UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Perceiving dynamic environments : from surface geometry to semantic representation

Syed Farzad Husain

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesi Doctoral

Perceiving Dynamic Environments: From Surface Geometry to Semantic Representation

Syed Farzad Husain

Directors:
Babette Dellen (RheinAhrCampus der Hochschule Koblenz)
Carme Torras (Institut de Robòtica i Informàtica Industrial, CSIC-UPC)

July 2016

Abstract

Robots have to perceive human environments in order to work effectively in domestic and assistive contexts. They need to recognize objects and identify human actions to properly interact with their surroundings. Nowadays, the environment is primarily captured using cameras that deliver color and depth images. Cues obtained from such images are the building blocks on which perception applications are developed. For example, appearance models are used for detecting objects and motion information is extracted from images for recognizing actions. However, given the complex variations of domestic and assistive settings, extracting a set of robust visual cues becomes harder here than in other contexts.

In this thesis we develop a hierarchy of tools to improve the different aspects of robot perception in human-centered, possibly dynamic, environments. We start with the segmentation of single images and then extend the developed techniques to handle videos. Next we develop a surface tracking approach that incorporates our video segmentation method. Afterwards we address the higher-level tasks of semantic segmentation and recognition of scenes. Finally, we focus on action recognition in videos. The introduction of Kinect-style depth sensors is relatively new and their usage in the field of robotics cannot be tracked back more than half a decade ago. Such sensors permit acquiring high-resolution color and depth images at a low cost. Given this opportunity, we dedicate most of our work to the exploitation of the depth information obtained with such sensors, thereby advancing the state-of-the-art in computational perception.

The thesis is conceptually divided into two parts. In the first part, we address the low-level tasks of segmentation and object tracking in depth images. In many cases, depth information provides a better disambiguation of surface boundaries between different objects in a scene than their color counterpart. We exploit this information in a novel depth segmentation scheme that fits quadratic surface models on different surfaces in a competitive fashion. We further extend the method to the video domain by initializing the segmentation results and surface model parameters from the previous frame for the next frame. In this way, we successfully create a video segmentation algorithm, in which the labeling of each surface becomes coherent over time. We also devise a particle-filter-based tracker that uses depth data to track a surface. The tracker is made more robust by combining it with our video segmentation approach.

The segmentation results serve as a useful prior for high-level tasks. In the second part of the thesis we deal with such tasks including (i) object recognition, (ii) pixel-wise object class segmentation, and (iii) action recognition. We propose (i) to address object recognition by creating context-aware conditional random field models. We show the importance of the context in object recognition by modeling geometrical relations between different objects in a scene. These relations are captured as potential edges in a graph. We perform (ii) object class segmentation using a convolutional neural network. The network is trained to minimize a pixel-wise cross entropy loss. We introduce a novel distance-from-wall feature and demonstrate its effectiveness in generating better class proposals for objects that are close to the walls. The final part of the thesis deals with (iii) action recognition. We propose a 2D convolutional neural network extended to a concatenated 3D network that learns to extract features from the spatio-temporal domain of raw video data. The network is trained to predict an action label for each video.

In summary, several perception aspects are addressed with the use of depth information where available. Our main contributions are (a) the introduction of a depth video segmentation scheme, (b) a graphical model for object recognition, and our proposals of deep learning models for (c) object class segmentation and (d) action recognition.

Acknowledgements

I would like to firstly thank both my advisors Babette Dellen and Carme Torras, to whom I owe this opportunity. I would also like to thank everyone who supported me during this long journey, and in particular Sergi Foix, Anais Garrell, Michael Villamizar, Bernat Joseph, Edgar Simo and Leonel Rozo for providing me with technical support and motivation.

This work has been sponsored by the following:

- A PhD scholarship.
- A three-month stipend to visit the Autonomous Intelligent Systems Group at Bonn University.
- Projects GARNICS (FP7-ICT-2009-4-247947), CINNOVA (201150E088), PAU+ (DPI2011-27510), IntellAct (FP7-ICT2009-6-269959) and TextilRob (201550E028).
- My grit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> The real voyage of discovery consists not in seeking new landscapes but in having new eyes.
>
> *Marcel Proust*

The concept of industrial robots that are automatically controlled, re-programmable and possess a multipurpose programmable manipulator as defined by ISO 8373 became a reality in the early 20^{th} century. The inception of industrial robots revolutionized the idea of mass production by minimizing human intervention. The automation of several tasks has led to significant improvements in production by reducing manufacturing costs and time. For example, a factory in China replaced 600 human workers with 60 robots along with a fivefold reduction in manufacturing errors and an increase in production of over 250 percent[1]. Given such advantages and luxuries provided by robots, it is no surprise that the traditional confinement of robots to industrial domains has gradually begun to make its way towards the domestic environments. Owing to the fact that robots need to be built, programmed and maintained, the demand in the job market is gradually shifting from repetitive work towards innovative and creative skills.

Cleaning the house, finding objects, assisting the elderly to increase their autonomy, and surveillance are some of the daily life, daunting chores that, if automated, could ease the lives of many modern humans. One possible solution is the inculcation of robots in human-assistive environments. However, robots which have to assist in domestic environments face a different set of perceptual challenges, as compared to industrial setups. This is because the robots in general have to deal with complex and deformable objects, pose uncertainties, and weakly structured environments. The safe and successful interaction of the robot with its environment in these scenarios relies on correct perception.

[1] http://spectrum.ieee.org/automaton/robotics/industrial-robots/chinese-unmanned-factory-replaces-humans-with-robots

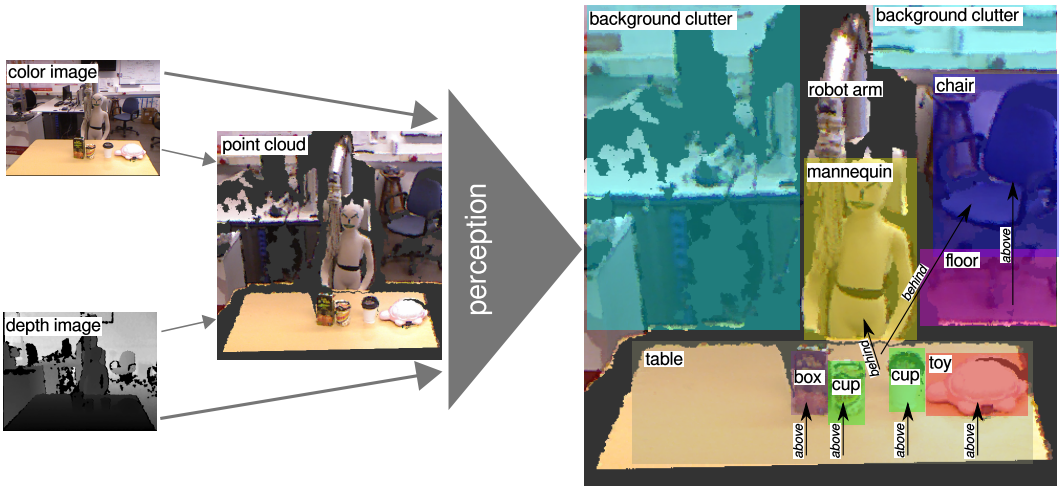

Figure 1.1: Understanding the semantics of an indoor scene is necessary for a robot to perform a task autonomously. Color image, depth image and the point cloud of an indoor environment are used to create a semantic representation. Visual cues are extracted after segmenting the scene and analyzing the individual segments. The relative positioning such as a chair-above-floor, cup-above-table, chair-behind-table are the cues that remain consistent in different environments thereby important for perception.

Perception is the ability to become aware of something through the senses. In the context of robotics, perception implies autonomous understanding of the environment that a robot has to deal with (Apolloni et al., 2005; Yan et al., 2013). To serve this purpose, a color image is the most basic form used as an input. A color image captures a scene with discrete color values at each pixel thereby giving a dense color representation. Other than the color values, the arguably most informative feature is the relative object-distances. The distance values give a more realistic perspective of the pose of the objects in a scene. The object-distances are usually represented by a depth image which encodes a distance value at each pixel.

A classical approach to obtaining the distance values is through passive stereo vision (Saxena et al., 2007). However, several pre-processing steps are required which becomes time-costly. Recently, quick depth perception at a low cost became possible with the introduction of structured-light-based sensors such as the Microsoft's Kinect. The Kinect sensor gives registered high resolution color and depth images. Such devices have received a lot of attention in the development of perception-related applications in the field of robotics.

Despite having a color and depth representation of a scene, perceiving it still requires bridging a rather huge gap between the pixel intensities and the scene semantics. Image segmentation and recognition are the most fundamental intermediates to bridge this gap. However, these tasks require coping with multiple challenges. Figure 1.1 illustrates a setup for perceiving different objects, necessary for autonomous tasks. Visual cues are obtained by extracting different surfaces through segmentation. The scene semantics such as different objects placed on the surface of a table can be obtained from the features of segmented surfaces and their relative positioning within the scene.

Image *segmentation* is performed by partitioning the image into multiple meaningful

segments. These segments assist in analyzing the image and serve as a prior for other higher-level tasks such as object detection. Obtaining the desired segmentation result is often difficult because of factors such as unclear surface boundaries, lighting conditions and background clutter. The simplest way to achieve segmentation is by clustering image pixels while maintaining some spatial connectivity pattern (Dhanachandra et al., 2015). However, this approach requires predefining the number of clusters and may lead to over-segmentation or under-segmentation. Histogram based segmentation is a way that avoids predetermining the number of clusters. The idea is to compute a histogram from all the image pixels and use its peaks and valleys to identify the image segments (Bonnet et al., 2002). Another approach for image segmentation is based on an undirected graphical model (Shi and Malik, 2000).

A closely related objective to segmentation is semantic labeling also known as object class segmentation. The goal is to divide an image into regions such that all the pixels from the same semantic class are assigned the same label. Semantic labeling can be thought as a next step to segmentation where each segment carries a class label. This is usually achieved by creating a graphical model (Lempitsky et al., 2011; He and Zemel, 2009; Ladicky et al., 2009; Gould et al., 2009). An image is pre-segmented and each segment is represented by a node in the graph. Afterwards, the parameters of the graph are learned from a set of features extracted from the segments and their relations with the neighbors such that a global loss function is minimized. The learned model is used to predict the labels for the test scenes.

Applications having temporal dynamics such as human-computer interaction, surveillance, and augmented reality require perceiving the scene over time. *Object Tracking* is a key component in these applications. However, building a robust tracker is challenging. Camera motion, frame rate, clutter, occlusions, illumination inconsistencies are some of the factors that worsen the performance of a tracker. The objective of tracking is to estimate the pose of an object by updating the parameters of a motion model. Usually tracking algorithms are either kernel based (Shen et al., 2010; Comaniciu et al., 2003; Kwon et al., 2009) or silhouette based (Isard and Blake, 1998; Bertalmio et al., 2000).

At the core of visual perception lies image *recognition*. It is a method for determining whether an image contains a particular type of content. It plays a central role in problems as diverse as content-based image retrieval, scene understanding and optical character recognition. The goal is to determine the level of similarity between features extracted from the images (Deselaers et al., 2008). Commonly used hand-crafted features include color histogram, Gabor filters (Park et al., 2002) for texture analysis and shape descriptors (Bober, 2001). The techniques from image recognition can be borrowed for recognizing actions in videos as well. This is because videos can be treated as a bunch of stacked images. However, there is also the added temporal dimension that cannot be ignored. This leads to an additional objective of extracting trajectories in the temporal domain that are relevant to a particular action. Optical flow has shown to be the most effective feature in the temporal domain (Wang and Schmid, 2013).

Contrary to hand-crafting features, deep learning is an approach that attempts to learn features at multiple levels of abstraction from datasets of images and videos. Recently, He et al. (2015) proposed a deep network architecture that surpassed human-level performance for visual recognition. Methods based on deep learning also yielded state-of-the-art results for recognizing actions in videos (Simonyan and Zisserman, 2014a; Tran et al., 2015).

Figure 1.2: In-house datasets are created using (a) Microsoft's Kinect camera and (b) PMD camera.

The availability of low-cost depth sensors has led to a growing interest in analyzing depth images. Depth data has the immediate advantage that the background clutter can be easily removed and is invariant to texture and lighting conditions. Additionally, depth data often shows a better disambiguation between different surfaces. This opens new possibilities for tackling perception problems more efficiently. In this thesis we investigate the usage of depth data for segmenting and tracking surfaces, which are otherwise hard using the cues from texture.

Our focus is on the perception of human assistive environments. We address robot perception which encompasses several key challenges. This includes image and video segmentation using depth data, tracking with depth data, object recognition, object class segmentation with RGB-D images and action recognition in color videos. We achieve our goals by devising new algorithms, handpicking features and using the advanced deep learning techniques. Figure 1.2 shows the Microsoft's Kinect and the PMD camera used for creating in-house datasets for our experiments.

1.1 Contributions

We organize the contributions into six groups: (1) depth image and video segmentation, (2) surface tracking and grasping, (3) segmentation-aware tracking, (4) object recognition, (5) object class segmentation, and (6) action recognition.

1. **Depth image and video segmentation.** We propose an approach to segment 3D point clouds into geometric regions. Unlike existing works that rely on local depth discontinuities to segment the depth data, our approach uses globally defined surface models, which makes it less sensitive to local changes. Our segmenter is more robust to noise levels as we show in the experiments. We also introduce a mechanism in the segmentation to adjust to changing input data along a video, leading to stable, temporally coherent, and traceable segments. We test the method on a large variety of data acquired with different range imaging devices, including a structured-light sensor and a time-of-flight camera. This work was published as parts of (Dellen et al., 2013; Husain et al., 2015).
2. **Surface tracking and grasping.** We develop a particle-filter-based tracking algorithm that uses depth data obtained from Microsoft's Kinect sensor to track

a surface in real-time and is robust to noise and partial occlusions. We show the applicability of our approach by implementing a system that continuously updates the pose of a gripper, until it has the moving object within its reach, and afterwards grasps the object. Using the depth data only makes our tracker independent of the texture. We demonstrate several scenarios in which our texture-independent tracker keeps on tracking, whereas other texture-based trackers fail. This tracker was published in (Husain et al., 2014a).

3. **Segmentation-aware tracking.** Problems with depth data arise, if a surface boundary disappears when two surfaces have a physical contact. In such a scenario, the tracker can get confused and stop tracking the desired surface. In order to palliate this deficiency, we combine our previously described video segmenter with our tracker to create a more robust version that is able to track better when the boundary of the desired surface is not clearly disambiguated. This enhanced tracker was published in (Husain et al., 2014c).
4. **Object recognition.** We present a supervised learning approach for the recognition of objects from 3D point clouds using Conditional Random Fields, a type of discriminative, undirected probabilistic graphical model. Owing to the fact that different objects are likely to be placed at a certain position relative to each other, such as a keyboard in front of a monitor, an improvement in the detection accuracy by exploiting these relations when compared to the object features alone is shown. Our method allows for learning and inference from unorganized point clouds of arbitrary sizes and shows significant benefit in terms of computational speed during prediction when compared to a state-of-the-art approach based on constrained optimization. This work was published as part of (Husain et al., 2014b).
5. **Object class segmentation.** We present a deep learning approach for semantic segmentation of indoor-scenes. Various strategies using convolutional neural network models are investigated. Multiple models are designed to exploit different characteristics of the data. This includes feature learning at a coarse to fine level and the introduction of a new feature that we call *distance-from-wall*. This feature is fed as an input to the network. By combining these modalities, state-of-the-art semantic segmentation results are obtained on the challenging NYU v2 dataset. This work was published in (Husain et al., 2016b). Our semantic segmentation approach was later used in a method that combines bottom-up object discovery and semantic priors for producing generic object candidates in RGB-D images (Martín García et al., 2016). These last two works stem from a collaboration with Prof. Dr. Sven Behnke, from the Autonomous Intelligent Systems group at the University of Bonn.
6. **Action recognition.** We finally tackle the problem of recognizing human actions from color videos. A 2D convolutional neural network model which was originally designed to recognize single image content is extended to 3D through concatenation. The network learns to extract features from the spatio-temporal domain of video data by training directly on the raw video content. Starting with a 2D convolutional neural network allows us to exploit a pretrained network as a descriptor that yielded the best results on the largest and challenging ILSVRC-2014 dataset. Experimental results on commonly used benchmarking video datasets demonstrate that our results are state-of-the-art in terms of accuracy and com-

putational time without requiring any preprocessing (e.g., optic flow) or a priori knowledge on data capture (e.g., camera motion estimation), which makes it more general and flexible than other approaches. This work was published in (Husain et al., 2016a).

1.1.1 Publications

The following is a list of the publications derived from this thesis:

Babette Dellen, **Farzad Husain** and Carme Torras. Joint Segmentation and Tracking of Object Surfaces in Depth Movies along Human/Robot Manipulations. In *International Conference on Computer Vision Theory and Applications*, 2013.

Farzad Husain, Babette Dellen and Carme Torras. Recognizing Point Clouds using Conditional Random Fields. In *International Conference on Pattern Recognition*, 2014.

Farzad Husain, Adrià Colomé, Babette Dellen, Guillem Alenyà and Carme Torras. Realtime Tracking and Grasping of a Moving Object from Range Video. In *International Conference on Robotics and Automation*, 2014.

Farzad Husain, Babette Dellen and Carme Torras. Robust Surface Tracking in Range Image Sequences. *Digital Signal Processing*, 2014.

Farzad Husain, Babette Dellen and Carme Torras. Consistent Depth Video Segmentation using Adaptive Surface Models. *IEEE Transactions on Cybernetics*, 2015.

Farzad Husain, Hannes Schulz, Babette Dellen, Carme Torras, and Sven Behnke. Combining Semantic and Geometric Features for Object Class Segmentation of Indoor Scenes. In *IEEE Robotics and Automation Letters*, 2016.

Farzad Husain, Babette Dellen and Carme Torras. Action Recognition based on Efficient Deep Feature Learning in the Spatio-Temporal Domain. In *IEEE Robotics and Automation Letters* , 2016.

German Martín García, **Farzad Husain**, Hannes Schulz, Simone Frintrop, Carme Torras and Sven Behnke. Semantic Segmentation Priors for Object Discovery. In *International Conference on Pattern Recognition*, 2016.

1.2 Thesis Overview

We include a chapter to lay some groundwork for the techniques used in this thesis. Afterwards, we group the work done conceptually into six major chapters as done in our contributions. Each of these chapters has an introductory section, several sections explaining different techniques, and a summary of the work. Finally the last chapter provides the concluding remarks for the thesis. The breakdown of the thesis is as follows:

Chapter 2: Overview. This introductory chapter presents some of the tools from computer vision and machine learning used in our work.

Chapter 3: Depth Image and Video Segmentation. This chapter presents a segmentation approach for depth data using quadratic surface models. We introduce a mechanism to segment a single depth image which is also extended to segment depth videos. This chapter is largely based on our 2013 VISSAP paper and our 2015 Transactions on Cybernetics journal paper.

Chapter 4: Surface Tracking and Grasping. This chapter introduces a particle-filter-based tracking algorithm that uses depth data to track a surface in real-time. Additionally, an implementation of a system to continuously update the pose of a gripper based on the tracked result is given. This tracker refers to our 2014 ICRA paper.

Chapter 5: Segmentation-Aware Tracking. In this chapter we build a more robust tracker by combining it with our video segmentation approach. This refers to our 2014 DSP journal paper.

Chapter 6: Object Recognition. This chapter introduces a supervised learning approach for the recognition of objects from 3D point clouds using Conditional Random Fields. The work refers to our learning model used in our 2014 ICPR paper.

Chapter 7: Object Class Segmentation and Discovery. In this chapter, we look into the problem of object class segmentation and discovery. A deep learning model is presented for semantic segmentation based on different object classes of indoor-scenes. The work presented in this chapter refers to our 2016 RA-L journal paper. The results are used to enhance a bottom-up object discovery method. This work is a collaboration with German Martín García, Hannes Schulz, Simone Frintrop, and Sven Behnke from the University of Bonn. German Martín García was the primary developer for the object discovery part.

Chapter 8: Action Recognition. This chapter focuses on recognition of human actions from color videos. We first give an overview of the approach and elaborate the related work in that field followed by the detailed description of our model. The work presented in this chapter refers to our 2016 RA-L journal paper.

Chapter 9: Concluding Remarks. This chapter summarizes the efforts we made and elucidate where our work stands with regards to recent developments. We also draw conclusions from the thesis and possible directions for future research work.

Chapter 2

Technical Preliminaries

In this thesis we devise several techniques for the advancement of the state-of-the-art in perception problems. We use the necessary tools from computer vision and machine learning to achieve our goals. This chapter briefly gives a rough overview of these tools as an introduction to their usage in this dissertation.

We will begin with an explanation of the affine model in Section 2.1.1 followed by a description of the Histogram of Oriented Gradients (Dalal and Triggs, 2005) in Section 2.2.1 and Persistent Feature Histogram (Rusu et al., 2008a) in Section 2.2.2. Finally in Section 2.2.3 we will discuss artificial neural networks. We divide the techniques into two categories. First are the dynamic models, whose parameters are updated after each time step. Second are the hand-crafted and learned features which are fed to a recognition system. Table 2.1 provides a list of different techniques used in our thesis.

2.1 Dynamic Models

We refer to the dynamic models as the ones that are updated according to the changing input. These models are primarily used for low-level tasks such as segmentation and tracking. We have used two such models. First is the adaptive surface fitting that we

Table 2.1: List of techniques used in this thesis and their usage in each chapter. Column header C stands for chapter so that C3 refers to Chapter 3.

Type	C3	C4	C5	C6	C7	C8	Technique
Dynamic Models	✓		✓		✓		Surface fitting
		✓	✓				Affine model (Kwon et al., 2009)
Descriptors				✓			HOG (Dalal and Triggs, 2005)
				✓			PFH (Rusu et al., 2008a)
					✓	✓	Convnet 2D (Sermanet et al., 2014)
						✓	Convnet 3D (Tran et al., 2015)

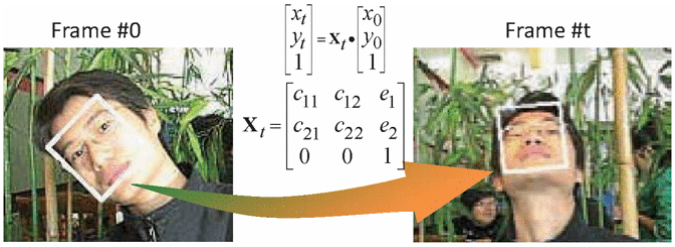

Figure 2.1: Illustration of affine motion tracking. $(x_0, y_0, 1)^T$ and $(x_1, y_1, 1)^T$ are the homogeneous coordinates of pixels in object regions at frame 0 and frame t, respectively; X_t is the affine transformation to be obtained by the tracking process. Figure reproduced from (Li et al., 2012).

have devised. This model is described in Chapter 3. The second is the widely used affine model which is explained in the next section.

2.1.1 Affine Model

An affine transformation is a linear 2D geometric transform that is used to map homogeneous pixel coordinates from (x_0, y_0) to (x_t, y_t), i.e.,

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = X_t \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}, \tag{2.1}$$

where

$$X_t = \begin{bmatrix} c_{11} & c_{12} & e_1 \\ c_{21} & c_{22} & e_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

providing six degrees of freedom corresponding to the six matrix elements in X_t and having the composite effects of rotation, scale, shear and translation. The motion estimation problem becomes a minimization of a distance measure between the template and the target by estimating the six parameters. Affine models are often used in surface tracking for establishing an approximate correspondence between the tracked surface in a target image plane and a template. Figure 2.1 illustrates an example of face tracking from (Li et al., 2012).

2.2 Descriptors

We will first explain two feature point descriptors, i.e., the Histogram of Oriented Gradients (HOG) and the Persistent Feature Histogram (PFH). These descriptors can be thought as simple priors as they do not carry any task specific information and describe the local object appearance and shape information. Afterwards, we will give an overview of the convolutional neural networks for image data whose output can be used as a descriptor for recognition-related tasks.

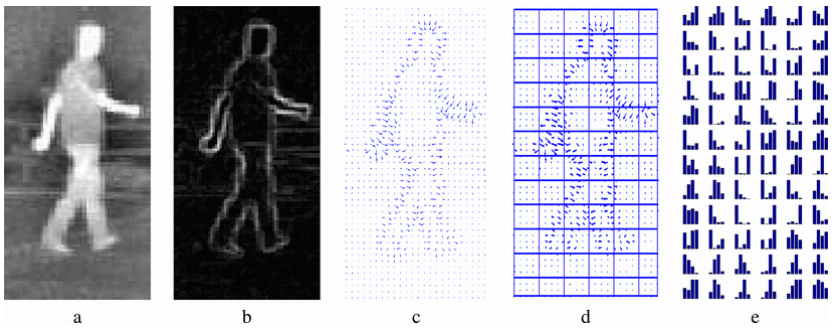

Figure 2.2: Computing the HoG features, (a) original image, (b) gradient norm, (c) gradient orientation, (d) cell splitting and (e) histogram computation. Figure reproduced from (Bertozzi et al., 2007).

2.2.1 Histogram of Oriented Gradients

HOG is a feature descriptor widely used for object detection in images. It describes the objects in a way that is robust to variations in appearance and illumination. The main idea behind the HOG descriptor is to calculate the distributions of the gradient orientations in small and connected image regions. Afterwards, the distributions are concatenated. Operating in small image regions brings invariance to geometric and photometric transformations. The HOG descriptor-based classification can be divided into the following steps:

1. Split the image into small spatially connected regions (cells).
2. Calculate a weighted local 1-D histogram of gradient directions over the pixels of each cell.
3. Group adjacent cells into blocks and normalize the histograms.
4. Accumulate the normalized histograms within a detection window (HOG descriptor).
5. Train a classifier such as an SVM using the descriptor.

Figure 2.2 taken from (Bertozzi et al., 2007) shows an example of computing the HOG descriptor for a human shape.

2.2.2 Persistent Feature Histogram

The PFH descriptor is used to describe a set of points belonging to a 3D point cloud, in a way that is invariant to camera pose, noise and sampling density. It is computed using the pairs of points p_i and p_j, and their estimated normals n_i and n_j that lie within a predefined neighborhood of a key point. The histogram computation process is carried out by selecting a source point p_s and a target point p_t. The source point has a smallest angle between the associated normal and the line connecting the points. The PFH consists of a histogram of four different measurements calculated for these point pairs, i.e.,

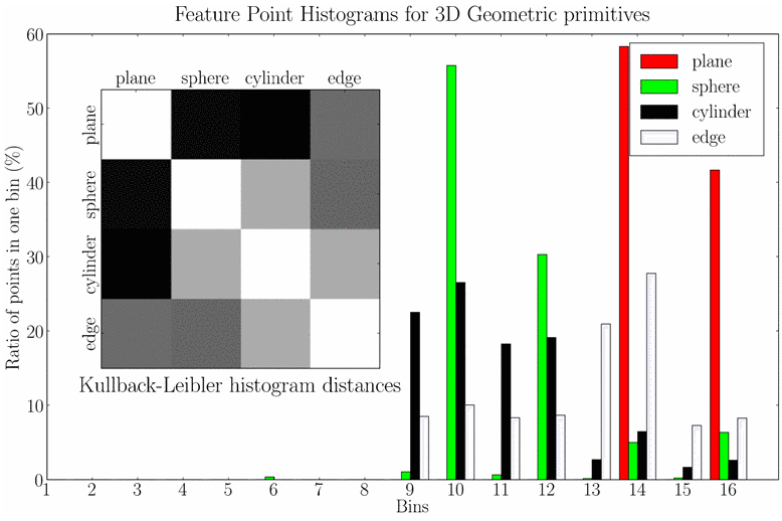

Figure 2.3: Persistent Feature Histogram for query points located on different geometric surfaces along with the color coded Kullback-Leibler distances. Figure reproduced from (Rusu et al., 2008a).

$$
\begin{aligned}
f_1 &= \langle v, t \rangle, \\
f_2 &= ||p_t - p_s||, \\
f_3 &= \langle n_s, p_t - p_s \rangle / f_2, \\
f_4 &= \text{atan}(\langle w, n_t \rangle, \langle u, n_t \rangle),
\end{aligned}
$$

where $v = (p_t - p_s) \times u$ and $w = u \times v$. These measurements encode the intrinsic geometric relations between pairs of points and their local surface normals. Figure 2.3 shows the PFH computed for 3D points sampled from different surfaces which, as can be seen, permit distinguishing the surfaces quite clearly.

2.2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are directed acyclic graphs and have recently shown to be the best performing computation tool for some computer vision tasks. This is credited to the availability of huge amounts of data along with the significant increase in computing power, which enabled training of complex networks with over 50 million parameters.

Each layer inside an ANN is composed of several computational units, termed "neurons", which are hooked together so that the output of neurons at layer l becomes the input of neurons at layer $l+1$, i.e.,

$$
a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}), \tag{2.3}
$$

where $W^{(l)}$ is the weight matrix of layer l, $b^{(l)}$ is the bias term and f is the activation function. The activation for layer l is denoted by $a^{(l)}$. Training an ANN requires learning W and b for each layer such that a cost function is minimized. Formally, given a training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ of m training examples, the weights W and

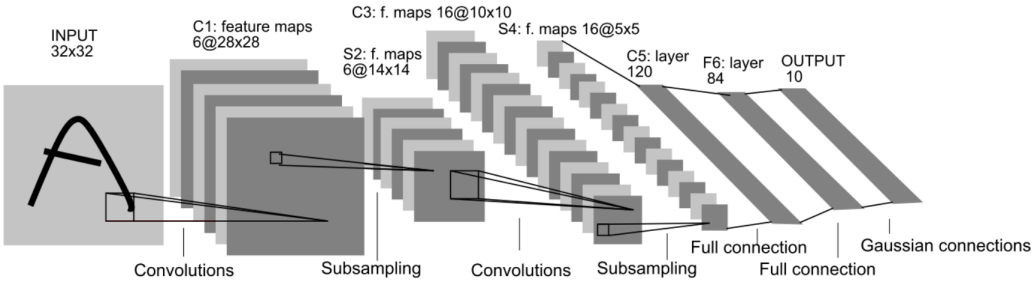

Figure 2.4: A Convolutional Neural Network (CNN) architecture for character recognition. The network contains alternating convolution and subsampling operations. Convolution operators reduce the number of parameters because the computational units within a feature map share the weight matrix. Subsampling operators reduce the size of the feature maps thereby increasing the position invariance of the convolved filters. Figure reproduced from (Lecun et al., 1998).

bias b need to be determined that will minimize the cost, i.e., difference between the desired output y and the actual output $f_{W,b}(x)$. The cost function for one training example is defined as:

$$J(W, b; x, y) = \frac{1}{2}||h_{W,b}(x) - y||^2, \tag{2.4}$$

where, $h(x)$ gives the activations of the last layer.

The minimization is done iteratively using a gradient descent approach which involves the computation of partial derivatives of the cost function with respect to the weights and updating the weights accordingly. One iteration of gradient descent updates the parameters W, b as:

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial}{\partial W^{(l)}} J(W, b), \tag{2.5}$$

$$b^{(l)} = b^{(l)} - \alpha \frac{\partial}{\partial b^{(l)}} J(W, b). \tag{2.6}$$

The backpropagation algorithm is used to compute the partial derivatives of the cost function.

ANNs that employ convolution operators are also known as convnets or Convolutional Neural Networks (CNNs). These networks are composed of various alternating convolutional layers and subsampling layers, often followed by a few fully-connected layers. The advantage of using convolutions in an ANN is that it exploits the 2D structure of an image and becomes easier to train because of fewer parameters. Figure 2.4 shows an example of a CNN designed for character recognition.

2.3 Summary

In this chapter we had a quick glance at the tools that are later used for developing our own techniques. These tools include the affine transformation model used in Chapters 4 and 5, the Histogram of Oriented Gradients and the Persistent Feature Histogram used in Chapter 6, and finally the Artificial Neural Networks used in Chapters 7 and 8.

Chapter 3

Depth Image and Video Segmentation

During human or robotic manipulations, we face the challenge of having to interpret a large amount of visual data within a short period of time. The data from the sensors needs to be structured in a way that makes task-relevant visual information more accessible. The recognition of objects and scene context in a temporally consistent manner plays here a central role. Segmenting different surfaces in a temporally coherent manner can serve as an important prior for such recognition related tasks.

In this chapter we introduce a scheme to segment depth data into non-overlapping surface patches. This includes a method to segment a single depth image and using the segmentation results as a prior for segmenting upcoming frames, thereby leading to a coherent video segmentation.

3.1 Introduction

Recent advances in 3-D sensing have revived the interest in depth data segmentation and extended their possible field of application to depth video segmentation. Time-of-flight (ToF) cameras and commercial structured-light devices (Kinect) allow acquiring depth images in real-time at a quality suitable for feature extraction, object recognition, 3-D reconstruction, tracking, mobile robot navigation and parts identification.

However, depth image segmentation is an area in computer vision that is less mature than, e.g., color segmentation, and if we extend the idea of segmentation to maintain *consistency* in a depth image sequence, then this field is almost unexplored. By consistency we mean that the same surface should get the same label in subsequent frames. Very little work has been done in this regard (Parvizi and Wu, 2008; Ghobadi et al., 2007; Jiang et al., 1999).

If we consider segmentation of a single depth image then most segmentation algorithms focus on either particular surface shapes (Ghobadi et al., 2007; Han et al., 1987), or segment the depth data using local surface descriptors measuring surface orientation (Pulli and Pietikainen, 1993), planarity (Hegde and Ye, 2011), curvature

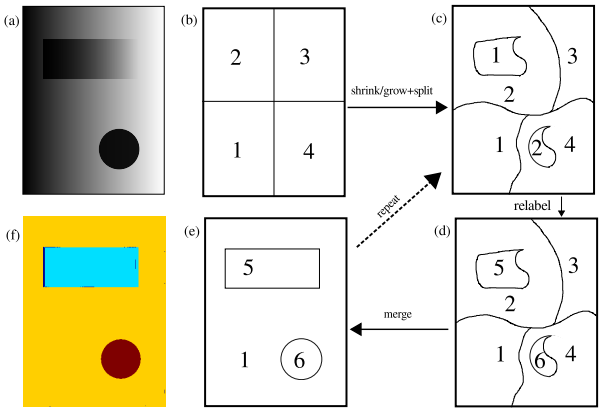

Figure 3.1: Schematic illustrating the main mechanisms driving the segmentation procedure. (a) Input depth data, (b) initial configuration of segments, (c) segments after growing/shrinking/splitting, (d) segments after relabeling, (e) segments after merging and (f) actual segmentation result obtained using our algorithm. Processes (c $\rightarrow$ d $\rightarrow$ e) are iterated until the final result is obtained.

(Bab-Hadiashar and Gheissari, 2006) or depth probability distribution (Parvizi and Wu, 2008). Other algorithms rely on statistical inferences for making decisions during segmentation (Han et al., 2004). This makes these algorithms slower and unfeasible for real-time applications. Comparisons of different depth image segmentation algorithms can be found in (Min et al., 2004; Powell et al., 1998; Hoover et al., 1996).

Depth video segmentation poses different demands than single depth image segmentation. Even small changes in camera position, or motions within a scene, can cause significant changes in the position and orientation of the surfaces from one frame to the next. For this reason, it is disadvantageous to work with a precisely defined segmentation based on local descriptors such as surface normals. Instead, we draw the segmentations from globally defined quadratic surface models which compete for label assignments within a local neighborhood, thus depending on the relative configuration of surfaces in the scene. In a related work, Leonardis et al. (1997) used superquadric models for segmenting a scene using a recover-and-select paradigm (Leonardis et al., 1997). First, the data is partitioned into a predefined number of small areas (splitting), which are then merged depending on the superquadric fitting error. While this approach yields satisfactory results for single images, it cannot be easily extended to video because the solution is obtained through progressive merging. Hence decisions cannot be revoked. This however is a fundamental requirement when searching for a method that can adapt to changing data, which fails to be met by standard split-and-merge approaches (Lakaemper and Latecki, 2006; Rhee et al., 2012).

This shortcoming motivated us to develop a split-and-merge method in which splitting and merging mechanisms are active at all times during the application of the procedure. This way, wrong splits can be revoked, and merges be undone, giving rise to two competing processes. Convergence is characterized by a constant number of segments, which can thus be easily determined. However, for this strategy to be successful, the competitive decision process for label assignment during the split-and-merge process has to be made dependent, at least partly, on relative values instead of absolute

thresholds. The main mechanisms driving the segmentation of the proposed approach are illustrated in Fig. 3.1, allowing for the dynamic growing, shrinking, removing, and adding of seed points instead of only growth and removal of predefined regularly spaced seeds on an image grid as in (Leonardis et al., 1997).

Furthermore, our algorithm does not follow the conventional tracking approaches where a predefined motion model (constant velocity/acceleration model) is used. Segments are solely analyzed based on their extrinsic surface geometry and temporal coherence. We use an adaptive model fitting approach for clustering unlabeled points, which makes the algorithm flexible enough to accommodate non-rigid transformations as well. In our method, the shape and position of the segments are continuously adapted to the data such that the surface model fitting error is minimized. During the video segmentation, we enforce the following supporting constraints in addition to the already introduced split-and-merge mechanisms shown in Fig. 3.1:

1. Each segment has to stay connected.
2. A segment can only be merged with those segments that have been in its proximity since the time of its creation.
3. A segment cannot be smaller than a minimum size (predefined in 'number of pixels').
4. A segment cannot move further than a maximum distance from one frame to the next.

These constraints play a central role in inducing consistency between segmented surfaces with the passage of time.

3.2 Related Work

Single Depth-Image Segmentation

Many algorithms have been proposed for single depth-image segmentation (Hoover et al., 1996; Jiang et al., 2000a; Jiang, 2000; Checchin et al., 1997; Koster and Spann, 2000; Frigui and Krishnapuram, 1999; Gotardo et al., 2004; Enjarini and Graser, 2012; Oehler et al., 2011; Wallenberg et al., 2011). Most of these methods use detection of local depth discontinuities to segment the depth data. Jiang et al. (2000b) proposed a single-depth-image-segmentation method which segments each horizontal scan line into quadratic curves. Afterwards, the longest curve segment is chosen as a seed and used to perform grouping based on quadratic functions, which also allowed handling curved surfaces. The widely used gPb/UCM hierarchical segmentation for color images (Arbelaez et al., 2011) is applied to both color and depth images and the segmentation results are linearly combined to generate a soft segmentation mask of the scene in (Ren et al., 2012). There are also different learning approaches using labeled 3D scan data for segmentation (Silberman et al., 2012; Anguelov et al., 2005). Collet et al. (2011) made some high-level shape assumptions to discover different structures in the scenes.

Leonardis et al. (1997) used superquadric models for segmenting a scene using a recover-and-select paradigm. One of the drawbacks of using superquadrics is the larger number of iterations ($\sim$ 15 in this case) needed to estimate the model parameters using Levenberg-Marquardt algorithm. In our approach we use rather simple quadratic surface models for segmenting and tracking depth data. Furthermore, in our method,

splitting and merging mechanisms are constantly competing, which allows earlier decisions to be revoked in subsequent iterations.

However, single depth-image segmentation has different demands than depth-video segmentation since it does not require that partition consistency is maintained across frames. Instead, the foremost goal of single depth-image segmenters is to segment the image with high accuracy and to resolve very small structures. For video segmentation, the resolution of small structures is of less importance, since the main purpose is the stabilization of the segment labels along the video.

Video Segmentation

To the authors' knowledge, little work has been done using depth information as the primary vision cue for joint segmentation and tracking. Parvizi and Wu (2008) performed multiple object tracking using an adaptive depth segmentation method. Time-of-flight depth was used to segment each frame independently by finding the connected components based on an absolute depth distance measure. The segments of adjacent frames were then associated with each other using a depth histogram distribution. However, this depth segmentation method is rather simple and does not partition the data into distinct surfaces. As a consequence, boundaries defined by changes in 3D shape (curvature) cannot be detected, which constitutes a major difference in comparison to our method. In addition, each movie frame is segmented from scratch. In the case of surface segmentation, this can be rather costly. Furthermore, the temporal consistency of the segmentations will degrade with increasing clutter in the scene.

Seeding and subsequent growing of segments has previously been proposed in (Jiang et al., 1999). An erosion of the previous segmentation is applied to generate seeds for the new frame. For region growing, orthogonal distance of a point to a plane is checked. The method has been tested with planar surfaces only, because the initial segmentation is restricted to planar surfaces. Wang and Liu (2012) obtained seeds by using a positive motion capture method followed by region growing.

Lopez-Mendez et al. (2011) performed upper body tracking of a human using a depth sensor (Microsoft Kinect). The technique is limited to human beings only, as they use a prior model of the human body. The RGB-D data from Microsoft Kinect is also used in (Pieropan et al., 2013) for object detection and tracking. Planar surfaces are detected in the depth image and afterwards, surfaces connected to the planes are discovered by computing the differences in color and depth values. To detect the same object along time, a tracker that uses histogram of HSV values is employed. Pauwels et al. (2013) proposed model based segmentation and tracking of rigid objects using cues from both color and depth data.

Joint segmentation and tracking has previously been performed mostly for color image sequences (Aksoy et al., 2011; Deng and Manjunath, 2001; Patras et al., 2001; Wang, 1998; Wang et al., 2009a; Grundmann et al., 2010). Many methods for color-video segmentation usually perform independent segmentations of each frame and then try to match segments (Deng and Manjunath, 2001; Patras et al., 2001; Wang, 1998; Grundmann et al., 2010). This is problematic because segmentations have to be computed from scratch for every frame, which has consequences on both the computational cost and the temporal consistency of the results. For cluttered scenes, the partition of the segmentation tends to change from one frame to the next, and temporal coherence of the segmentations is prone to be impaired because of this effect.

In another work, segmentation and multi-object tracking were performed simultaneously using graphical models (Wang et al., 2009a). Observed and hidden variables of interest describing the appearance and the states of objects are jointly considered and used to formulate the objective as a Markov random field energy minimization problem. Different from our method, depth measurements do not enter the framework, and objects are defined based on their 2D appearance alone. Also, objects of interest are defined in the first frame and are then tracked along the sequence. While the method delivers convincing results, energy minimization is computationally expensive and efficient optimizations would have to be developed to make the approach more practical.

Aksoy et al. (2011) segmented color images by finding the equilibrium states of a Potts model. Consistency of segmentations obtained along the movie and the tracking of segments were achieved through label transfer from one frame to the next using optic flow information. This way, the equilibrium states in the current frame could be encountered more rapidly. The resulting segments represent regions of uniform color and usually do not coincide with the object surfaces in a geometric sense, which we would desire for our system. The solutions found in (Aksoy et al., 2011) cannot be easily adapted to our problem, because color segmentation and depth segmentation are inherently different problems. Surfaces cannot be defined based on local properties only, which increases the difficulty of the problem considerably.

3.3 Algorithm

We describe a method for segmenting a movie of depth images into surface patches. A surface patch should contain a smooth surface, which can be planar or curved. Segments that describe the same surface should carry the same label along the sequence, corresponding to a tracking of the objects in the scene.

In our method, a coherent segmentation of a current frame is obtained by recycling the segmentation which has been computed for the preceding frame. Only the first frame of the sequence has to be segmented from scratch, simply because it has no predecessor. To obtain the segmentation of the first frame, we need to choose a starting point. Normally, two or four segments are taken initially to cluster the entire scene. Then the algorithm is iterated over the initial frame while updating the segmentation. Afterwards, we re-use the segmentation obtained for frame F^t to find the segmentation of frame F^{t+1}, and so on, where t denotes the frame number. The main components of the algorithm are presented in Fig. 3.2. The algorithm consists of four main steps:

A. Seeding/growing/splitting: Segment regions, labels, and the respective surface models of frame F^t are transferred to frame F^{t+1}. For each pixel, the depth predicted by the surface model of a segment is compared with the measured depth values found in the respective (preliminary) segment region. If the depth difference is below an adaptive threshold ψ, it is considered a seed for the segment (shrinking). The remaining points are unlabeled. Using seed points only, the surface models are re-estimated. The seeds are grown by assigning non-seed points in a competitive way (growing/splitting) to the neighboring surface that predicts the depth value of the point with the smallest error.

B. Relabeling: The assignment of new labels during the growing phase does not guarantee that the segments defined by the new labeling represent connected

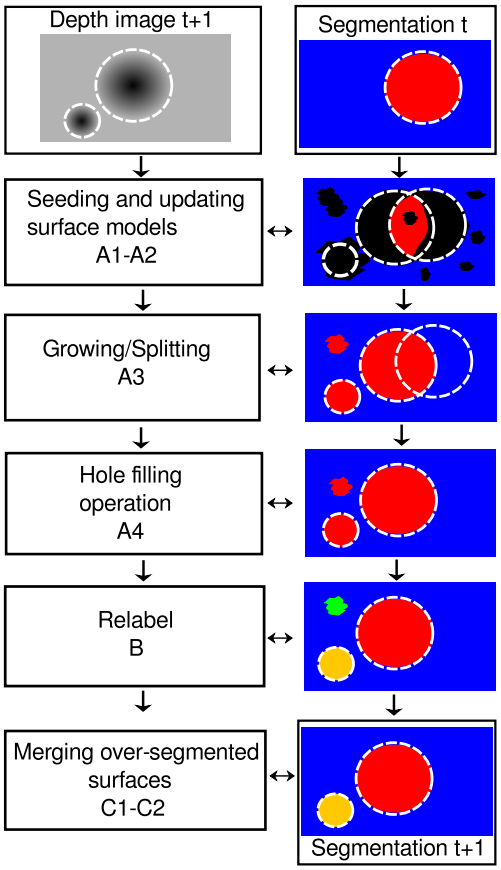

Figure 3.2: Schematic illustrating the steps of our algorithm. Undefined areas are colored white, which usually correspond to the object contours. Black color represents the unlabeled regions in the segmentation.

components. This is resolved by determining all connected components for a label and relabeling all components but the largest.

C. Merging: Neighboring segments are merged if they can be described approximately by the same surface model. Because only direct neighbors are considered, segments that have been split from a larger segment during the relabeling phase cannot be merged back with the same segment during a single iteration. This way, conflicts between splitting and merging do not arise, which facilitates convergence.

D. Iteration: Steps A-C are repeated until the number of segments stabilizes, indicating convergence. During video segmentation, the depth data is updated with the new movie frame at each iteration. For obtaining the initial segmentation, the procedure is applied repeatedly for the same frame. Note that while within a single iteration revokes are not allowed, both splitting and merging decisions can nevertheless be undone in the next iteration, which allows the method to adjust to changing data.

A pseudo code of the method including all mandatory steps of the algorithm is provided in Fig. 3.3. Parameter values used for segmentation are provided in Table 3.3 in

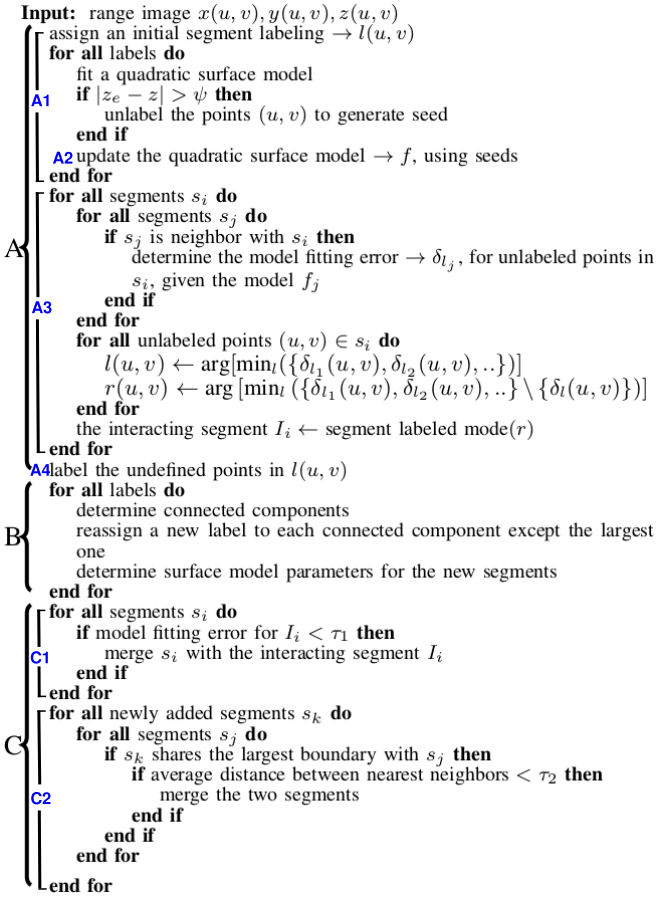

Figure 3.3: Pseudo code describing a single iteration. Steps A1-C2 are in accordance with the illustration in Fig. 3.2. More details about the algorithm can be found in Section 3.9.

the Section 3.9. Detailed calculation of variables such as ψ and specific implementation choices - taking into account special characteristics of the depth data used (e.g. holes) - are also provided in the Section 3.9.

Since merging is a particularly delicate/critical operation in the process, some more details provided here as follows.

During the merging step C1, we do not test for all possible merging combinations. Instead, during the growing step A3, the second-best label assignment for the unlabeled points is determined for each segment s and the most frequent label defines the interacting segment for s, representing the candidate for a potential merge. Then, segments are merged dependent on a threshold τ_1. New segments added during step B are treated separately and tested for merges using a threshold τ_2 with the segment with which they share the largest boundary in step C2. In step B, new labels are assigned and at this particular moment this may represent a temporally incoherent event. This incoherence is needed in order to account for new objects that may enter or leave the scene. However if a new segment is falsely created it can be merged back during step C2 or in successive iterations during step C1.

3.4 Image Acquisition

For in-house data, a Microsoft Kinect sensor along with the Kinect package of ROS (Robot Operating System) was used to acquire sequences of depth images F^1,...,F^t, F^{t+1},... F^{t+n} for different scenarios. Each frame contains a matrix of size $w \times h \times 3$, where w and h are the spatial dimensions of the image grid. Each point in the grid stores the values of the three coordinates (x, y, z) in the Euclidean space. The algorithm is implemented in Matlab.

3.5 Model Fitting

A quadratic surface model $f_j(x, y)$ of the form

$$z_e = f_j(x, y) = ax^2 + by^2 + cx + dy + e \quad , \tag{3.1}$$

with surface parameters a, b, c, d, and e is fitted to each segment s_j by performing a Levenberg-Marquardt minimization of the mean square distance of the measured depth points $z(u, v)$ from the estimated model depth $z_e(u, v) = f_j[x(u, v), y(u, v)]$. The chosen model type allows modeling of planar and curved surfaces, e.g., cylinders and spheres. The iterative solver (Levenberg-Marquardt minimization) enables us to use the solution obtained for the preceding time step to initialize the current minimization procedure. This way, fewer iterations (~ 4) are required to reach the minimum. For the initial time step we set the starting state of the iterative solver to zero.

3.6 Performance Measure

We use the segmentation covering measure, as described in (Arbelaez et al., 2009) to determine how closely the segmentation results match the ground truth segmentation. The ground truth (column (d) of Fig. 3.6, 3.7 and column (c) of Fig. 3.8, 3.12) was created by initializing with our segmentation result and afterwards correcting the wrong labels manually. The ground truth may for this reason contain a minor bias towards our method. For one frame, the segmentation covering measure of a machine segmentation S by a human segmentation S' is defined as

$$C(S' \rightarrow S) = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} O\left(R, R'\right) \quad , \tag{3.2}$$

where N is the total number of pixels in the image, $|R|$ the number of pixels in region R, and $O(R, R')$ is the overlap between the regions R and R' defined as

$$O(R, R') = \frac{|R \cap R'|}{|R \cup R'|}. \tag{3.3}$$

Furthermore, for comparison of single depth-image segmentation with other methods, we use the evaluation framework of (Hoover et al., 1996).

3.7 Results

The algorithm was tested with several depth movies showing human and robot manipulations of objects. Additionally, the results which are acquired and used to generate all the figures and their corresponding datasets are available at http://www.iri.upc.edu/people/shusain/datasets_segmentation.html.

Segmentation of Initial Frame/Convergence

The segmentation of the first frame is obtained by iterating the algorithm over a single depth image. Figure 3.4 shows the convergence result for an indoor scene. The numbers represent the corresponding unique label for each segment. The color image (Fig. 3.4(a)) is only shown for illustration but not used in the procedure. We observe the formation of stable segments after a few updates (Fig. 3.4(e)). The convergence plot is shown in Fig. 3.4(f). It can be observed that after a certain number of iterations the segmentation covering measure became stable. For the images tested, we arrived usually at a stable configuration within 10 iterations. In order to terminate the iterative process at a stable point, we needed to define a stopping criterion. We compute a leakage factor ℓ between consecutive iterations, i.e.,

$$\ell = \operatorname{sgn}\left(N_{t+1} - N_t\right) \sum_{i=1}^{k} \left| n_i^{t+1} - n_i^t \right|, \tag{3.4}$$

where N is the number of segments, n_i is the number of pixels in segment s_i and $k = \max(N_t, N_{t+1})$. When the leakage factor does not change anymore, the system has reached a minimum and the process can be terminated.

In Fig. 3.4 it can be seen that even though only a very small portion of the ball (spherical surface) is visible, its points are still clustered correctly. The algorithm further successfully resolved the depth differences at the boundaries between the table and the objects.

Figure 3.5 shows the segmentation result obtained with our method for a scene captured with a Laser Range Sensor (LMS-Z210 by Riegl) of size 444×1440 pixels, by the pattern theory group of Brown university (Brown, 2015). The segmentation was initialized in the same way as in Fig. 3.4, and 50 iterations were required. Despite the complex structure of the scene, our method successfully segmented it into meaningful object surfaces. Particularly compared to (Han et al., 2004) we obtained less oversegmentation.

Depth Video Acquired using a Structured-Light Camera

We report results on a set of sequences recorded with the Microsoft Kinect sensor. Firstly, we present results for a human hand rolling with its fingers a green ball forward and then backwards (see Fig. 3.6). In Fig. 3.6(a) and (b), selected calibrated color and depth images acquired with the Kinect are shown, respectively. Areas for which no depth value could be acquired are marked white. In Fig. 3.6(c), our segmentation results are shown. Fig. 3.6(d) shows the ground-truth segmentation for comparison. The segments are color-coded, where each color corresponds to a unique segment label. The ball and the hand are correctly segmented and tracked along the image sequence, even though the hand is changing its shape and the ball is changing its size during the

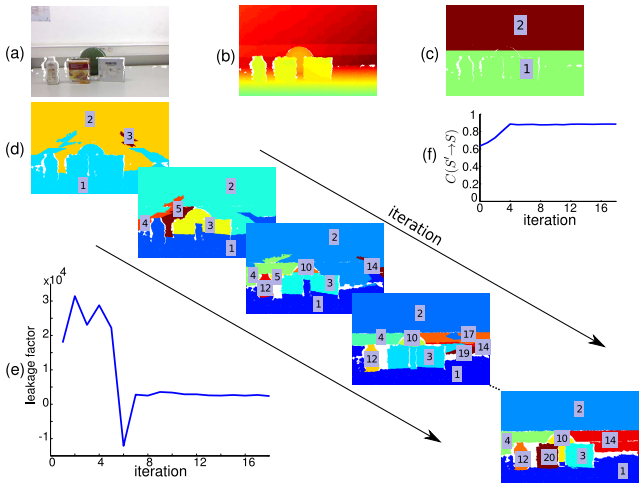

Figure 3.4: Convergence of a single frame to a stable solution for a scene containing spherical and planar surfaces. (a) Color image from Kinect, (b) depth image (Kinect), (c) initialization with just two segments, (d) segmentation results after iterating using our method and (e) the leakage factor after each iteration. The value of the leakage factor after convergence is ∼2366 in number of pixels. (f) Segmentation score, demonstrating convergence to a stable solution.

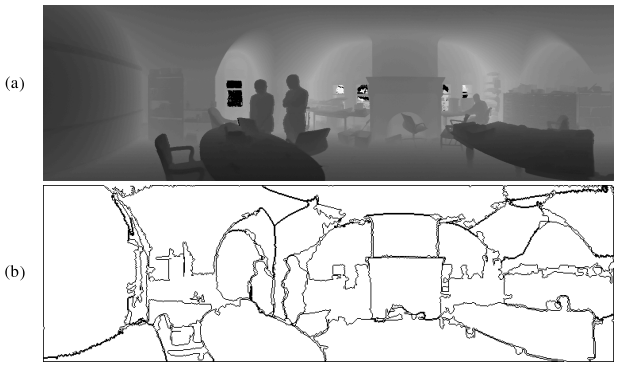

Figure 3.5: Segmentation result for a scene from the Brown depth dataset. (a) Range image and (b) our segmentation result.

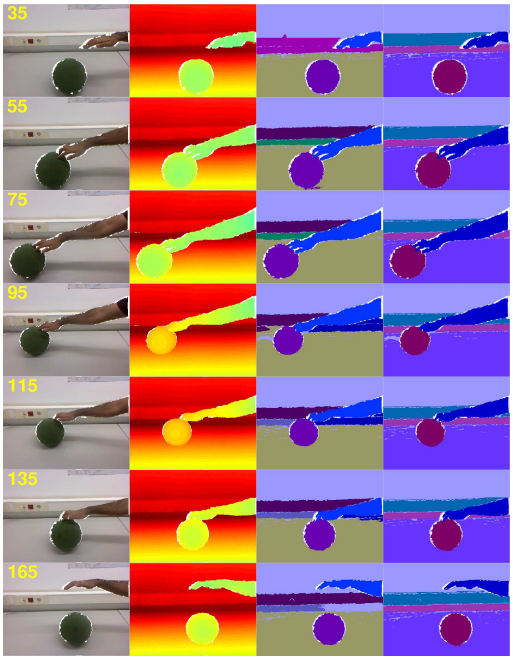

Figure 3.6: Hand rolling a ball. Undefined areas are colored white. (a) Color images from Kinect, (b) depth images (Kinect), (c) video segmentation results using our method and (d) the human segmentation used as ground truth.

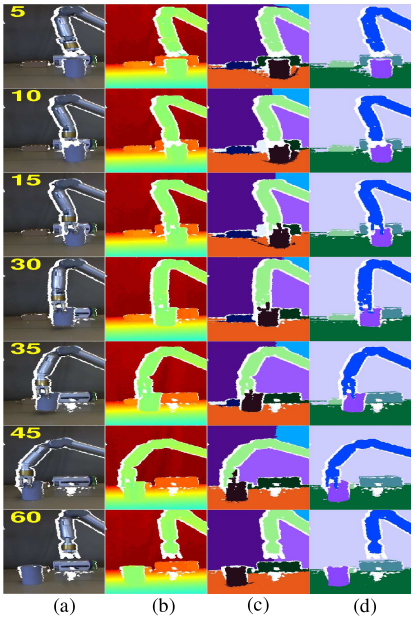

Figure 3.7: WAM robotic arm grasping and displacing a paper roll. Undefined areas are colored white. (a) Color images from Kinect, (b) depth images (Kinect), (c) video segmentation results using our method and (d) human segmentation used as ground truth.

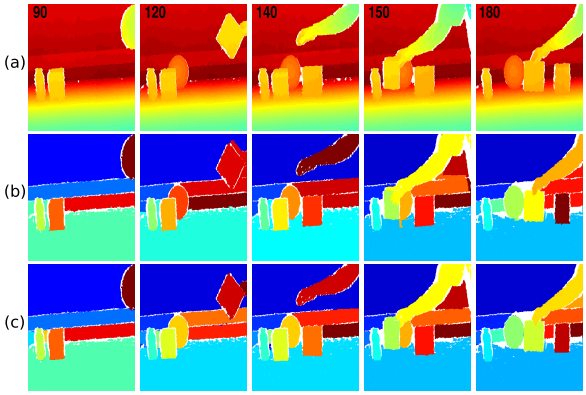

Figure 3.8: (a) Depth images, (b) machine segmentation and (c) human segmentation.

motion, so at least two different kinds of rigid transformations, i.e. translation and scaling, are taking place in this scene. Also the complete manipulator (hand plus arm) was not visible in the initial frame, but it is still segmented correctly later on.

We also show segmentation results for a movie where the robot arm grasps a paper roll and moves it to a new position (see Fig. 3.7). During the movement, objects in the background become occluded. Nevertheless, the sequence is correctly segmented and both the robot arm and the paper roll are tracked along the movie. The background gets oversegmented because it got disconnected. The method can further handle the non-rigid transformation of the robot arm. Since our method does not rely on any kind of predefined shape models, it allows accommodating non-rigid behavior of objects. Also it can be observed that the robot arm is cylindrical and the tissue role as well, hence the depth values at the point of their contact become very similar, but due to the enforcement of constraint 2 (see Section 3.1), the algorithm does not merge these two surfaces.

Figure 3.8 shows the depth images along with the corresponding machine and human segmentations of selected frames from a depth movie. The sensor pose was static, while different objects were manipulated. In this scenario, the human hand was entering and leaving the scene, displacing the objects rapidly, leaving little or no overlap of corresponding segments between consecutive frames. This naturally led to some temporal instabilities, so only the segmentation performance for single images can be evaluated, not the consistency along the movie. This movie will be used to compare the performance of the algorithm with those of different color-based video-segmentation algorithms in Section 3.8.

Another scenario in which multiple segments are tracked simultaneously is shown in Fig. 3.9. Here, a plant is manually displaced on top of a cluttered table. It can be observed that as the plant is being displaced, multiple segments are tracked jointly through the scene. The advantage of using depth over color in real world is quite obvious in this scenario, since segmenting and tracking plant leaves using only color with no a priori information becomes impossible.

We also applied our approach to a typical sequence from the RGB-D Object Dataset (Lai et al., 2011). The authors only provided the ground-truth mask of the object (the pitcher) placed on the turntable. Figure 3.10 shows selected frames of the segmented video sequence. It can be seen that the object on the turntable was successfully seg-

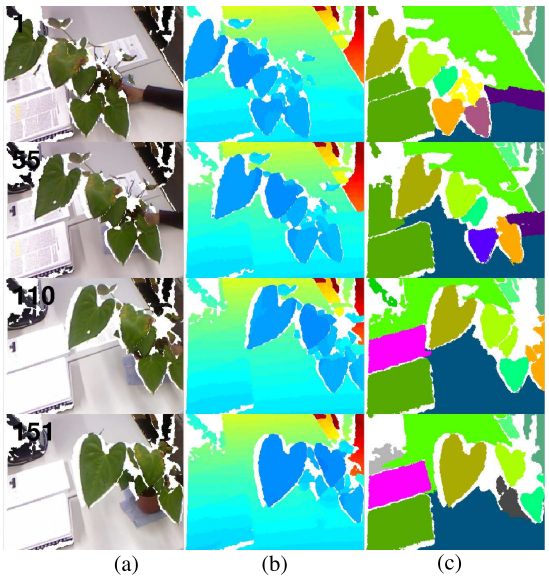

Figure 3.9: Plant being displaced. Undefined areas are colored white. (a) Color images from Kinect, (b) depth images (Kinect) and (c) video segmentation results using our method.

mented while being rotated.

Depth Video Acquired using a Laser Range Finder

The algorithm was tested on a scene of cluttered lab environment captured using odetics LADAR (Laser Detection and Ranging) camera. The dataset was obtained from the USF depth image database (USF, 2015). The robot moved quite rapidly in the scene leaving little overlap between consecutive frames. We had to iterate the algorithm three times over each frame to get consistent results. Fig. 3.11 shows selected typical intensity and depth images along with our segmentation results. The prism shaped surface and the ground floor kept the same label throughout the movie.

Depth Video Acquired using a Time-of-Flight Camera

Figure 3.12 shows the depth images along with the corresponding machine and human segmentations of selected frames from a depth movie. The data was obtained from the publicly available dataset of the Department of Robotics Systems at DLR (DLR, 2015). In this scenario, the scene is static and the camera is rotating 2° at each time step. We iterated three times over each frame in order to get more consistent results along the movie.

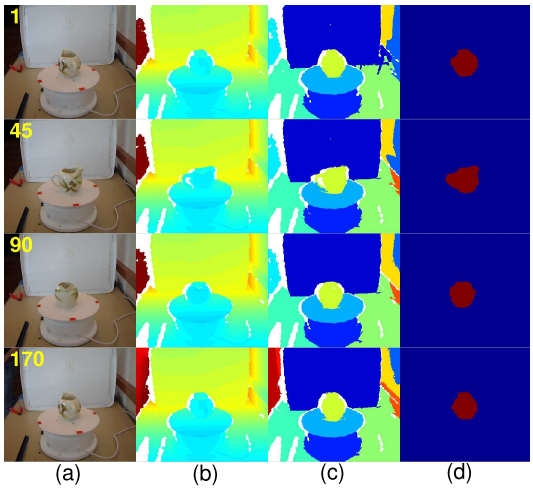

Figure 3.10: Pitcher rotating on a turntable from the RGB-D Object Dataset (Lai et al., 2011). Undefined areas are colored white. (a) Color images from Kinect, (b) depth images (Kinect), (c) video segmentation results using our method and (d) ground truth of the segmented pitcher.

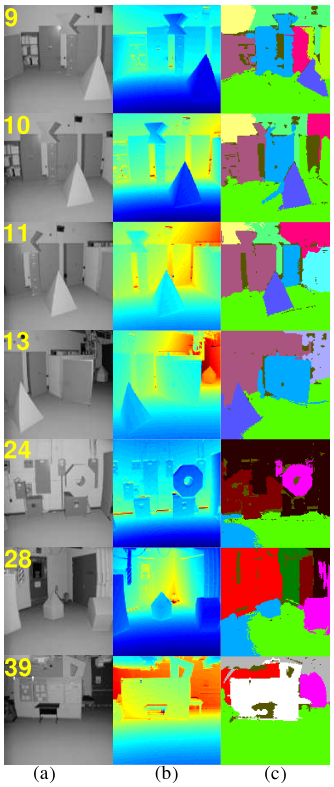

Figure 3.11: Data acquired with a mobile robot in a cluttered lab environment, using odetics LADAR camera. From the USF depth image database. (a) Intensity images, (b) range images and (c) video segmentation results using our method.

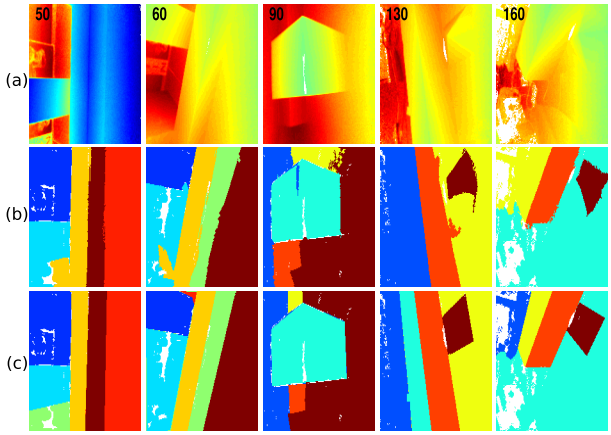

Figure 3.12: Segmentation results of a video obtained from a time-of-flight camera. (a) Depth images, (b) machine segmentation and (c) human segmentation.

3.8 Performance Evaluation and Comparison with other Methods

Comparative Evaluation of Single Range Image Segmentation

We report segmentation performance of our method for the ABW and the Perceptron datasets (USF, 2015) using the evaluation framework of (Hoover et al., 1996) and compare it to various single-depth-image segmenters (Hoover et al., 1996; Jiang et al., 2000a; Jiang, 2000; Checchin et al., 1997; Koster and Spann, 2000; Frigui and Krishnapuram, 1999; Gotardo et al., 2004; Enjarini and Graser, 2012; Oehler et al., 2011; Bab-Hadiashar and Gheissari, 2006) (see Table 3.1). For these datasets, we first used the provided training sequence (10 samples) to determine the optimal parameter values for ρ, τ_1 and τ_2. The datasets (ABW and Perceptron) contain only planar surfaces and the comparison has been done for planar segmentation only, hence we replace eq. 3.1 with the equation of a plane. We used a genetic algorithm to determine the parameter values which minimize the cost function,

$$cost = \frac{1}{M} \sum_{i=1}^{M} \left[1 - C_i(S' \rightarrow S)\right], \tag{3.5}$$

where M is the number of training samples. The remaining parameters were set as described in Section 3.9. Afterwards, we evaluate the performance with the provided test sequence (30 samples).

For the ABW dataset, our method showed better results in terms of

1. correct detection than WSU, OU, PPU and UA,
2. over-segmentation than WSU, UB, UE, UBP, UBham, RCA and UFPR/OSU,
3. under-segmentation than PPU, UA and UMel,
4. missed regions than WSU, OU, PPU and UA and
5. noise than WSU, UB, OU, PPU, UA and RCA.

For the Perceptron dataset, our method showed better results in terms of

1. correct detection than WSU,

Table 3.1: Average results of 16 segmenters on ABW and 11 segmenters on perceptron test set at 80% compare tool tolerance.

Algorithm	ground truth	correctly detected	over-segmented	under-segmented	missed	noise
ABW 30 test images						
USF (Hoover et al., 1996)	15.2	12.7 (83.3%)	0.2	0.1	2.1	1.2
WSU (Hoover et al., 1996)	15.2	9.7 (63.8%)	0.5	0.2	4.5	2.2
UB (Hoover et al., 1996)	15.2	12.8 (84.2%)	0.5	0.1	1.7	2.1
UE (Hoover et al., 1996)	15.2	13.4 (88.1%)	0.4	0.2	1.1	0.8
OU (Jiang et al., 2000a)	15.2	9.8 (64.4%)	0.2	0.4	4.4	3.2
PPU (Jiang et al., 2000a)	15.2	6.8 (44.7%)	0.1	2.1	3.4	2.0
UA (Jiang et al., 2000a)	15.2	4.9 (32.2%)	0.3	2.2	3.6	3.2
EG (Jiang, 2000)	15.2	13.5 (88.8%)	0.2	0.0	1.5	0.6
UBP (Checchin et al., 1997)	15.2	13.0 (85.5%)	0.6	0.3	1.0	1.3
UBham (Koster and Spann, 2000)	15.2	13.4 (88.1%)	0.4	0.3	0.8	1.1
RCA (Frigui and Krishnapuram, 1999)	15.2	13.0 (85.5%)	0.8	0.1	1.3	2.1
UFPR/OSU (Gotardo et al., 2004)	15.2	13.4 (88.1%)	0.4	0.1	1.2	1.7
GoD (Enjarini and Graser, 2012)	15.2	13.2 (86.8%)	0.3	0.2	1.1	1.8
UBonn (Oehler et al., 2011)	15.2	11.1 (73.0%)	0.2	0.7	2.2	0.8
UMel (Bab-Hadiashar and Gheissari, 2006)	15.2	12.8 (84.2%)	0.0	1.5	0.8	N.A
Our Approach	15.2	10.0 (66.0%)	0.3	0.8	3.0	1.9
Perceptron 30 test images						
USF	14.6	8.9 (60.9%)	0.4	0.0	5.3	3.6
WSU	14.6	5.9 (40.4%)	0.5	0.6	6.7	4.8
UB	14.6	9.6 (65.7%)	0.6	0.1	4.2	2.8
UE	14.6	10.0 (68.4%)	0.2	0.3	3.8	2.1
EG	14.6	10.5 (71.9%)	0.0	0.2	3.6	1.6
UBP	14.6	10.6 (72.6%)	0.2	0.6	2.6	2.0
UBham	14.6	11.2 (76.7%)	0.1	0.2	2.9	5.2
RCA	14.6	9.6 (65.8%)	0.7	0.2	3.7	3.6
UFPR/OSU	14.6	10.8 (74.0%)	0.1	0.1	3.4	2.0
GoD	14.6	10.7 (73.3%)	0.4	0.1	3.6	4.4
Our Approach	14.6	7.23 (49.5%)	1.9	0.3	4.7	4.4

2. under-segmentation than WSU and UBP,
3. missed regions than WSU and
4. noise than WSU and UBham.

Our method has a performance that is overall comparable to other segmenters, but cannot outperform highly accurate segmenters such as the USF method. This is because the resolution of small image structures is of less importance in video segmentation, since fine details cannot be stabilized in the video anyway due to noise and changes in the temporally varying data.

However, the use of adaptive surface models, which are hooked on global structures rather than local ones such as jump edges or local changes in surface orientation, makes our method less sensitive to small changes in the depth data, and therefore more robust to noise. We demonstrate this by comparing it to a standard depth segmenter from USF (Hoover et al., 1996), for different levels of noise (see Fig. 3.13) added to the 30 images of the ABW sample test data using the segmentation coverage measure

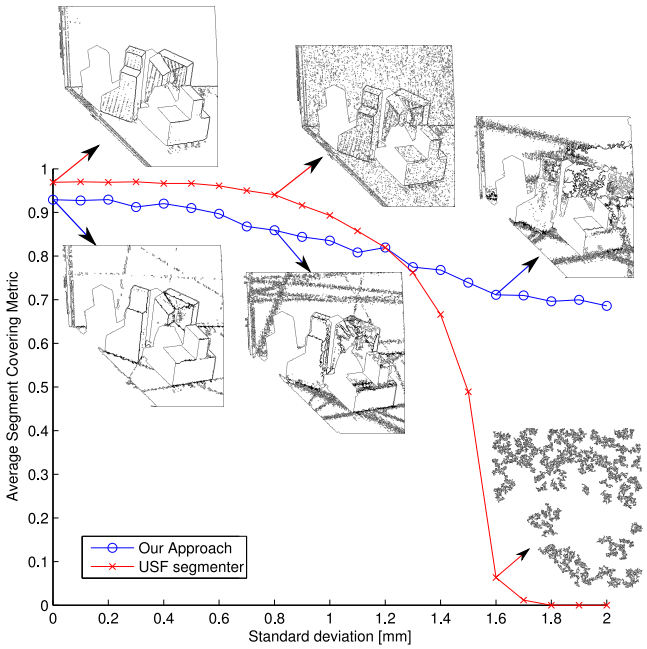

Figure 3.13: Average segment covering measure for the 30 images of the ABW test dataset, along with example results of depth image abw.test.8 for both our approach and the USF segmenter at different noise levels.

Table 3.2: Comparison of F-measure evaluation for boundary detection on nyu depth dataset.

Algorithm	**Range image segmentation from Table 1 in (Ren et al., 2012)**	**Our Approach**
F-measure (Arbelaez et al., 2011)	0.421	0.422

for evaluation. As can be seen in Fig. 3.13, for low levels of noise ($\sigma < 1.2$ mm) the USF segmenter performs better as it is able to delineate even very small image structures. However, for noise levels larger than 1.3 mm, the performance of the USF segmenter decreases rapidly, while our method only shows a slight decay, demonstrating its robustness.

We further evaluated our segmentation approach on the NYU depth dataset (Silberman et al., 2012) by calculating the F-measure using the boundary-detection evaluation criterion as proposed in (Arbelaez et al., 2011). It can be seen in Table 3.2 that we obtained a similar F-measure value as the approach of (Ren et al., 2012). Figure 3.14 shows results for selected images of the dataset.

We additionally tested our approach on the more recent Cornell (77 samples) (Anand et al., 2013) and UBonn (30 samples) (Oehler et al., 2011) datasets and obtained an average segmentation covering measure of around 60%. The ground truth labeling for the aforementioned datasets was supplied by the respective authors.

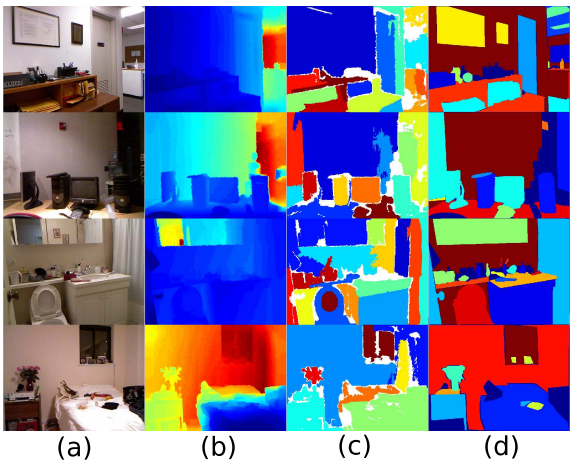

Figure 3.14: Selected images from the NYU depth dataset (Silberman et al., 2012). (a) Color images, (b) depth images, (c) segmentation results using our method and (d) human segmentation.

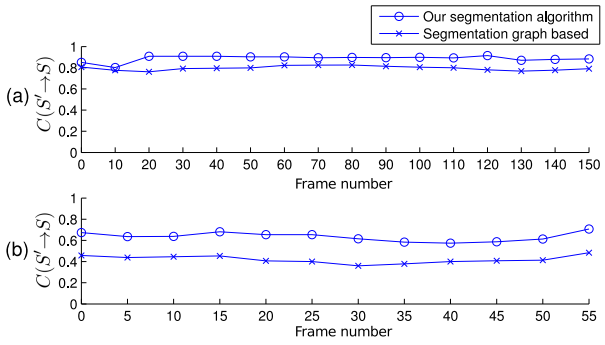

Figure 3.15: Segmentation score with respect to human segmentation for our segmentation and for color segmentation using (Grundmann et al., 2010), corresponding to (a) Fig. 3.6 and (b) Fig. 3.7.

Comparative Evaluation of Video Segmentation

Since there are no available benchmarks on depth video, we compare our method to two available color-video segmentation methods. Fig. 3.15 shows a plot of the segment covering measure for movies corresponding to Fig. 3.6 and Fig. 3.7. The measure is computed for our segmentation results and the ones obtained with a video segmentation algorithm based on color as described in (Grundmann et al., 2010). Not surprisingly, our method outperforms (Grundmann et al., 2010) for those scenes where object surfaces contain multiple colors, and obtain similar results if the surfaces possess a unique color.

Figure 3.16 shows the covering measure for the movies shown in Figs. 3.8, 3.10 and 3.12 computed for frames chosen at fixed time intervals. For comparison Fig. 3.16(a) also shows the measure for two different video segmentation algorithms based on color as described in (Abramov et al., 2012; Papon et al., 2012) and (Grundmann et al., 2010). It can be seen that our method outperforms the color-video segmenters. To obtain Fig. 3.16(b), we computed the covering measure for the pitcher only, since the

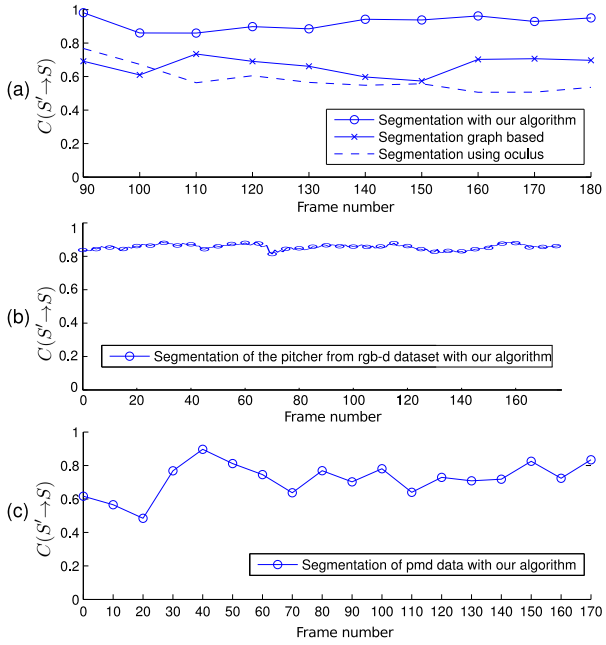

Figure 3.16: Segmentation score of the depth movies corresponding to (a) Fig. 3.8 with our approach, graph based (Grundmann et al., 2010) and oculus (Abramov et al., 2012; Papon et al., 2012), (b) Fig. 3.10 and (c) Fig. 3.12 with our approach only.

ground truth was provided only for this object (Lai et al., 2011). Overall, we observed that the proposed method allows segmenting videos while maintaining performance.

3.9 Algorithmic Implementation

The headers in this section are labeled according to the grouping A, B, C in the pseudo code in Fig. 3.3.

Depth segmentation of the first frame: An *initial segment labeling* $l^0(u, v)$ is defined as

$$l^0(u,v) = \begin{cases} 1 & \text{if } u \leq \lfloor h/2 \rfloor, \\ 2 & \text{otherwise} \quad , \end{cases} \tag{3.6}$$

where u and v are the indexes of the image grid and h is the frame height. The initial segment label matrix is multiplied with a binary mask B^0 defined as

$$B^0(u,v) = \begin{cases} 0 & \text{if } F^0(u,v) = 0, \\ 1 & \text{otherwise} \quad , \end{cases} \tag{3.7}$$

where the zeros (undefined points) correspond to the holes (undefined values) in F^0. Binary masks B^t for upcoming frames are defined in the same way. Steps A1-C2 (see below) are repeated for F^0 until a labeling l^t for the first frame is obtained.

In the following we assume that a labeling $l^t(u, v)$ has been already computed. Using this labeling we compute the current labeling of frame F^{t+1} as follows:

A1 Seeding: In order to update the segmentation grid, the first step should be to unlabel the points (u, v) that do not fit the surface. We achieve this by generating seeds. For each point (u, v) of frame F^{t+1}, we find the projected label $p = l^t(u, v)$ from the previous segmentation and define a seed labeling for F^{t+1} according to

$$m^{t+1}(u,v) = \begin{cases} p & \text{if } |z_e(u,v) - z(u,v)| < \psi_p, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

with $z_e(u, v) = f_p^t[x(u, v), y(u, v)]$ and

$$\psi_p = \sum_{(u,v)\in s_p^{t+1}} |z_e(u,v) - z| / (\rho n_p) \quad , \tag{3.9}$$

where s_p^{t+1} is the segment s_p^t projected onto the current frame F^{t+1}, n_p is the number of pixels in the area of s_p, and ρ is a constant. This defines a labeling $l^{t+1}(u, v) = p$ for all $(u, v) \in s_p^{t+1}$.

A2 Updating models: Now the surface model parameters need to be updated, so that they can model the current state of the segments. For each label j we obtain a surface model $f_j^{t+1}(x, y)$ for the current frame by applying the fitting procedure as explained in Section 5 to the seed of s_j, consisting of all the points (u, v) for which $m^{t+1}(u, v) = j$ holds.

A3 Growing/Splitting: Once we have updated the model parameters, we can determine the new labels of non-seed points. All points (u, v) with $m_j^{t+1}(u, v) = 0$ are grouped into connected components. For each connected component c_i, we search for seed points in the neighborhood R_1 of all boundary points (u, v), where R_1 is the radius of the disk used as a structuring element to dilate the connected components. If a seed point is found, its label is added to the list of potential labels $L_i = \{l_1, l_2, ..\}$ for c_i. For each label $q \in L_i$, we compute the distance

$$\delta_{l_q}(u,v) = |f_q^{t+1}[x(u,v), y(u,v)] - z(u,v)|. \tag{3.10}$$

For all $(u, v) \in c_i$, we can set

$$l^{t+1}(u,v) = \arg[\min_l(\{\delta_{l_1}(u,v), \delta_{l_2}(u,v), ..\})] \quad , \tag{3.11}$$

defining the labeling for the non-seed points. We also find the second-best fitting label for the group of non-seed points from each segment, i.e.,

$$r^{t+1}(u,v) = \arg\left[\min_l\left(\{\delta_{l_1}(u,v), \delta_{l_2}(u,v), ..\} \setminus \{\delta_{l^{t+1}}(u,v)\}\right)\right]. \tag{3.12}$$

We then define the *interacting segment* for segment s_j as

$$I_j^{t+1} = \text{mode}(r_j^{t+1}), \tag{3.13}$$

where the mode function determines the value that occurs most frequently.

A4 Hole filling operation: The undefined points that are present in the segmentation $l^{t+1}(u,v)$ due to the holes B^t in the previous frame F^t need to be filled in first in order to reduce oversegmentation due to addition of new segments in Step B.

First the edges of the depth image F_z^{t+1} are computed using the Canny operator and dilated using a disk of radius R_2 as a structuring element (SE), giving

$$\begin{aligned} E^{t+1}(u,v) &= canny\left\{F_z^{t+1}\right\} \quad , \\ E_d^{t+1}(u,v) &= E^{t+1}(u,v) \oplus SE. \end{aligned} \tag{3.14}$$

The dilated edges are used to disconnect undefined points in F^{t+1} which extend across different surfaces. Then each connected group of undefined points is assigned the label of the segment with which the largest boundary is shared. The current labeling is updated accordingly.

B Ensuring connectedness of each segment labeled $l^{t+1}(u,v)$ and assigning new labels: The assignment of new labels in Step A3 does not guarantee that the segments defined by the new labeling represent connected components (i.e., splitting). So we enforce Constraint 1 in this step. For each label j in $1,\ldots,k$, we find all points (u,v) for which $l^{t+1}(u,v) = j$. If the respective area is disconnected, we determine the connected components in the area. All connected components which do not define the largest component in the group are unlabeled. If an unlabeled connected component has a number of pixels greater than a constant κ it is assigned a new label and model parameters corresponding to its surface are estimated. This is how new surfaces are accommodated. Else it is assigned the label of the segment with which the largest boundary is shared. The current labeling $l^{t+1}(u,v)$ is updated accordingly.

C1 Label merging of *interacting segments*: The interacting segments I_j^{t+1} as determined in Step A3 are taken into account. We define a dissimilarity measure between the two segments s_i and s_j by estimating how well the surface model of segment s_i describes the depth data of segment s_j and vice versa. Let f_i be the surface model of segment label s_i, and f_j the surface model of segment label s_j. Then, we compute the fitting errors

$$\xi_{i/j} = \sum_{(x,y,z)\in s_i} |f_j(x,y) - z|/n_i \quad , \tag{3.15}$$

and

$$\xi_{j/i} = \sum_{(x,y,z)\in s_j} |f_i(x,y) - z|/n_j \quad , \tag{3.16}$$

where n_i and n_j are the number of pixels in segment s_i and s_j, respectively. If the sum of the two errors $\xi_{i/j} + \xi_{j/i}$ is less than a threshold τ_1 and Constraint 3 (see Section 3.1) is fulfilled, we assume that both segments model the same surface. Then the segments are merged and the surface model parameters are updated. After a segment is merged with another, it is no longer considered as interacting with other segments.

C2 Confirming new segments: In this step only the segments that were newly added in Step B are checked against the neighboring segment with which they share the largest boundary. Let s_i be the newly added segment and s_j be the segment with which it shares the largest boundary. We translate s_i and s_j such that their centroids are at the same position yielding s'_i and s'_j. Then we compute the average distance $\overline{\triangle}$ between nearest neighbors in s'_i and s'_j. If $\overline{\triangle} \leq \tau_2$, we assume that both segments belong to the same surface, so they are merged and the surface model parameters are updated.

D Steps A1-C2 are repeated for the next frame using $l^{t+1}(u, v)$ as initial labeling.

The choice of *initial segment labeling* is not fixed and could be initialized in any other configuration. Our choice is based on the fact that our focus is on modeling indoor environments and the background is usually divided in this manner (the front facing wall and the floor). This leads to an initial convergence in fewer iterations. In our experiments we needed to adjust only the control parameter R_2 according to the depth sensor used. The remaining parameters were kept constant.

Table 3.3 summarizes the definition of variables and constants used in our algorithm.

3.10 Summary and Future Work

In this chapter we have presented a novel algorithm for joint segmentation and tracking of object surfaces, defined by their geometric shape. Segments obtained for the first frame are used to initialize the segmentation procedure of the next frame, and so on. We tested the algorithm for several movies acquired with different depth sensors in cases where the sensor pose is static and others in which it is changing. Different scenarios were investigated including human and robot manipulations of objects. The algorithm allowed us to segment and track the main object surfaces in the scene, despite frequently occurring occlusions, limited resolution of the depth images, and shape changes of the hand and the robot gripper. However, some problems still remain. Occasionally, depth differences between surfaces are too small, resulting in assignment conflicts that cannot be resolved by the method as it is. For example, at boundary regions, where a hand or robot gripper touches an object and assimilates its shape, pixels are often assigned to the wrong surface temporarily, since the local depth differences are insufficient for recovering the true boundary line. A motion model could be used to predict the most likely poses of the surfaces in the upcoming frames, for example by using a particle filter. Currently we are exploring such a mechanism to resolve assignment conflicts.

We also provide a mechanism for generating new segments in addition to the ones that have been determined in the first frame, which is important in case new objects are entering the scene.

We provided a quantitative evaluation of the method using human-annotated ground truth. Obtaining ground-truth for video is however a very tedious procedure and thus poses us limits. Since there is no available implementation of a similar algorithm performing joint segmentation and tracking in depth space, we compared our method to two standard color-video segmentation algorithms (Abramov et al., 2012; Papon et al., 2012) and (Grundmann et al., 2010). We could show that our method outperformed color-video segmentation for the videos analyzed. However, this comparison may not be entirely fair, since we are using a different feature, i.e., depth, and not color.

Table 3.3: List of variables and constants

Variable	Definition
t	Time step
F	Frame
x, y, z	Coordinates in camera reference frame
u, v	Location indexes in image grid
h	Image grid height
B	Binary mask
s_i, s_j	Segments
l	Segment label
p	Projected segment label
n	Number of points
m	Segment seed
a, b, c, d, e	Surface model parameters
δ	Distance between measured and estimated depth
ξ	Mean model fitting error
c	Connected component
r	Second best label
I	Set of interacting segments
E	Edge image
E_d	Dilated edge image
ψ	Adaptive threshold for seeding
$\overline{\triangle}$	Average distance between nearest neighbors
Constant	**Value**
R_1	10 pixels
R_2	3 pixels (Kinect), 1 pixel (others)
ρ	1.7
κ	1000 pixels
τ_1	0.1 meters
τ_2	0.02 meters

We further evaluated the performance of our method with respect to single depth image segmentation. While our method has overall a performance comparable to other depth segmenters, it cannot outperform highly accurate techniques such as the USF method (see Table 3.1). This is because video segmentation has different demands than single depth-image segmentation. Our foremost goal is the stabilization of segments along the frames of the videos, for this reason, fine details have to be neglected. However, since our method uses global rather than local image criteria for segmentation, it is more robust to noise as compared, e.g., the USF method. The performance of the USF method rapidly deteriorates with the inclusion of Gaussian noise, as can be seen in Fig. 3.13, while the results obtained by our method are less affected by noise. We also measured our performance using the NYU depth dataset (Silberman et al., 2012) and obtained a similar performance as a recent segmentation approach (Ren et al., 2012).

The video segmentation algorithm may be used to extract important contextual scene information that can serve as a useful prior for tasks such as action recognition (Husain et al., 2016a). Fig. 3.17 shows a segment neighboring matrix for the entire depth movie corresponding to Fig. 3.12. The value of the matrix is one if the corre-

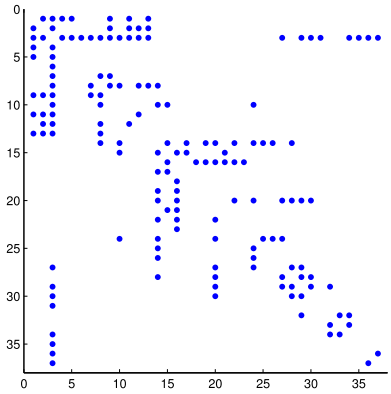

Figure 3.17: Segment relations matrix corresponding to the results shown in Fig. 3.12.

sponding two segments are neighbors. The depth movie contained a total of 180 frames. It can be observed that segment 3 was neighbor with most segments from almost the beginning until the end of the movie. Hence it can be deduced that this segment surface is modeling a permanent surface of the scene, which can only be the ground floor as the camera was rotating and the entire scene was changing except the floor.

The computation speed depends on the frame size used. Currently, it varies from ~ 0.45 seconds to process a frame of size 143×175 pixels to ~ 1 second to process a frame of size 430×282 pixels in Matlab on Intel Xeon 3.3 GHz processor. Our implementation mostly involves arithmetic operations on matrices. With an efficient C/C++ implementation of our method using multi-threading libraries, real-time performance might be in reach, which is one of our next goals. Since color information complementary to depth is provided by the Kinect camera, optic flow obtained from color images could be used to get a better estimate of the initial segmentation for the upcoming frame (Abramov et al., 2012). This might help minimizing the segment overlapping requirement between consecutive frames, and we plan to investigate this aspect in the future.

To our knowledge, the method presented in this work is the first algorithm for joint segmentation and tracking of geometrically defined object surfaces, described by quadratic functions, using depth, not color. Because the segmentation is grounded on depth values, only the geometric shape of the surface matters, leading to invariance with respect to other object features such as color or texture. We plan to use our method in the context of a learning-from-demonstration task (Agostini et al., 2011; Rozo et al., 2013). The method may also be relevant for tackling problems frequently encountered in industry which require handling of geometric objects, as for example bin picking.

Chapter 4

Surface Tracking and Grasping

In this chapter we introduce a tracking system which uses the depth data. We provide a brief description of the existing works and their shortcomings which motivated us to develop a novel method for tracking and afterwards using the tracked information for grasping moving objects. We demonstrate the applicability of the tracking system in realtime.

4.1 Introduction

In the field of robotics, many applications have been tailored towards servoing using visual information. The goal is to use information obtained from vision inside a servo loop to control a mobile manipulator. Visual servoing is broadly classified into two categories, i.e., image-based and position-based (Chaumette and Hutchinson, 2006). Most of the servoing tasks require the target to be stationary. Camera configurations such as eye-in-hand (Malis et al., 1999) and eye-to-hand (Kulpate et al., 2005) have been used. Tracking is often performed on the basis of color/grayscale images (Maeda et al., 2012; Song et al., 2011; Li et al., 2005). However, if the objects to be tracked are only weakly textured and do not contain distinctive color features, tracking may fail.

Recent advancements in range sensing technology for indoor scenes have lead to the development of a multitude of practical vision applications, however only little work has been done with respect to visual-servoing solutions based on range images. In this chapter, we present a novel approach for eye-to-hand, moving target, position-based servoing using depth as the only visual cue. This has the immediate advantage that the performance of the tracker is independent of the appearance of the objects in terms of color, and may thus generalize better to different scenarios. Tracking is achieved by a geometric particle filter on the affine group (Kwon et al., 2009). The respective, estimated affine transformation is applied to a bounding box placed in the range image, and a rigid transform is used to compute the measurement function. In this sense, tracking is entirely data driven, and no 3D object model needs to be used, neither for tracking nor for grasping. This also reduces the computational cost of the method. For the experiments, we use a Microsoft Kinect sensor calibrated to a Barret WAM arm

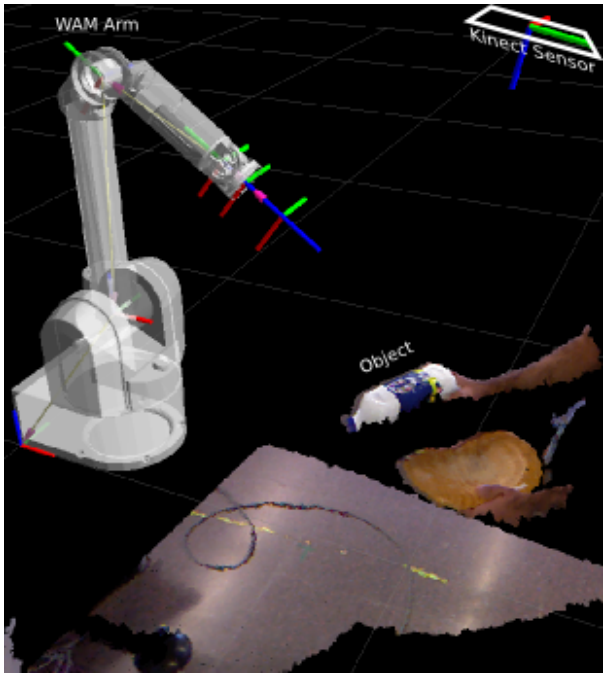

Figure 4.1: Illustration of the experimental setup containing the Barret WAM arm, the Kinect sensor mounted above the scene, and the 3D reconstruction of the scene, including the target object.

with 7-DoF as shown in Fig. 4.1. The output from the tracker is efficiently coupled with the pose of the end-effector. A smooth trajectory is created online from the pose of the object which controls the joint angles of the WAM arm, by using a robust inverse kinematics algorithm, as in (Colomé and Torras, 2012, 2015).

Potential applications of tracking include automatically picking up objects from a moving conveyor belt, incremental learning from demonstration, and human-robot interaction (Goodrich and Schultz, 2007).

4.2 Related Work

In the past, many different approaches have been developed for tracking and grasping of moving objects (Allen et al., 1993; Archibald and van de Panne, 1991; Kondak et al., 2001; Ge and Jie, 2007; Smith and Papanikolopoulos, 1995; Bing and Xiang, 2008; Benameur and Belanger, 1998; Lei and Ghosh, 1993; Hirota and Watanabe, 1990; Siradjuddin et al., 2012). Most of these methods use stereo images to compute the 3D pose for grasping. Matching scores required for tracking are usually computed from the color/grayscale features of the object. For example, in (Allen et al., 1993), the 3D pose of a moving object is computed using stereoscopic optic flow and used to control the motion of the robotic arm until interception and grasping is performed. The motion model employed to control the positioning of the robotic arm is restricted to planar trajectories, hence limiting the approach to grasping objects moving on a planar surface only. Similar constraints for motion in 2D have been employed in (Archibald and

van de Panne, 1991). In (Kondak et al., 2001), objects move on a conveyor belt in a straight line, and the precise trajectory of the objects could thus be provided.

Grasping a moving object using an eye-in-hand configuration has been performed in (Smith and Papanikolopoulos, 1995). However, the method presented therein is restricted to a single class of objects. Also the motion of the object is restricted to translations. A simulation of tracking and grasping a moving object based on a CAD model along with the dynamic selection of feature points has been presented in (Bing and Xiang, 2008). Based on the tracked results a real-time motion planning method is proposed. Simulation results have also been provided in (Benameur and Belanger, 1998) for an integrated sensing and actuation system for grasping a moving object.

A position-based and an image-based scheme for tracking and grasping problem have been proposed and compared using simulations in (Lei and Ghosh, 1993). However, here it is assumed that the pose of the object is already known.

They tracking and grasping setup described in this chapter closely resembles the one used in (Siradjuddin et al., 2012), but the tracking algorithm used in that work is rather simple. A blob detection algorithm based on the target image color is employed which can easily fail under varying appearance or lighting conditions.

Another possible approach to tracking using depth is to directly determine the 3D rigid transform that the tracked surface undergoes. This approach has been adopted in the OpenNI tracker for the Kinect sensor in the Point Cloud Library [1], where the 6 parameters of the rigid transform are predicted using a particle filter. Different to our approach, no motion model is employed, resulting in less efficient prediction of future states.

4.3 Problem Statement

Given an object moving in 3D space, we would like to reposition the end-effector of the manipulator continuously, until the distance between the moving object and the manipulator is smaller than a threshold. We assume that the object remains inside the workspace of the manipulator and it moves slow enough to be tracked and approached by the manipulator.

4.4 Object Surface Tracking

We track the object surface in the 2D image plane using a bounding box supplied by the user in the initial image, enclosing the pixel coordinates of the tracked surface template in the image plane. To determine the pose of the surface at each time instant t, we apply a geometric particle filter on the affine group, yielding an estimate of the 2D affine transform of the tracked surface in the image plane with respect to the initial coordinates, as proposed in (Kwon et al., 2009). The estimator uses a constant velocity model for the state dynamics i.e., the state update equation is expressed as

$$X_t = X_{t-1} e^{[a \ \log(X_{t-2}^{-1} X_{t-1}) + W_t]}, \tag{4.1}$$

[1] Available at: http://pointclouds.org/

where X is the 3×3, 2D affine transformation matrix, a is the autoregressive process parameter, and W is the Wiener process noise with covariance $Q \in \Re^{6\times6}$, owing to the 6 free parameters of the 2D affine group. The measurement equation is expressed as

$$y_t = h[X_t, I_{t=0}(P)] + v_t, \tag{4.2}$$

where h is the measurement function, P is the set of points representing the pixel coordinates of the tracked template in the image plane. I is the range image, i.e., for each $p \in P$, $I(p)$ gives the actual range value $(x, y, z)_p$ in the Euclidean space and v is the the Gaussian noise with covariance $R \in \Re^1$. Unlike (Kwon et al., 2009), we have defined the measurement function as

$$h(X_t, I_{t=0}(P)) = \left\| I_{t=0}(P) - I'_t(P'_t) \right\|_1, \tag{4.3}$$

where P'_t is the set of pixel coordinates obtained after transforming every $p \in P$ with X_t. Since we are using the range data, we cannot directly compute the difference between $I_{t=0}(P)$ and $I_t(P'_t)$ as it is the case of color/grayscale images (Kwon et al., 2009). This is because the range data provides the Euclidean distances relative to the camera pose, hence we first determine the rigid transform such that the distance between $I_{t=0}(P)$ and $I_t(P'_t)$ is minimized in a least square sense (Horn, 1987). $I'_t(P'_t)$ represents this set of points obtained after applying the rigid transformation.

The surface template $I_{t=0}(P)$ is periodically updated every five frames by computing the mean of the 3D shape that has been tracked during this interval. The particle filtering approach (Doucet et al., 2000) consists of the following main steps:

1. Sample $X_t^{(i)} \sim p(X_t | X_{t-1}^{(i)}, y_t)$
2. Compute weights $w_t^{(i)} = w_{t-1}^{(i)} \frac{p\left(y_t | X_t^{(i)}\right) p\left(X_t^{(i)} | X_{t-1}^{(i)}\right)}{\pi(X_t^{(i)} | X_{0:t-1}^{(i)}, y_{0:t})}$
3. Resample $X_t^{(i)}$ according to $w_t^{(i)}$

Detailed description for calculating the measurement likelihood $p\left(y_t | X_t\right)$ and the importance function $\pi(X_t | X_{0:t-1}, y_{0:t})$ is described in detail in (Kwon et al., 2009; Kwon and Park, 2010).

The affine transform encodes the deformation of a planar shape moving in 3D space and acquired under an orthogonal projection. In the case of non-planar surfaces with out-of-plane rotations, the affine transformation cannot determine exact point correspondences. However, extensive experimental results with non-planar surfaces have revealed that under weak-perspective assumption such a transformation can be approximated with a two-dimensional affine transform in the image plane. Previously, this assumption has been made in (Kwon and Park, 2010; Zhou et al., 2004; Li et al., 2007; Ross et al., 2008) for color/grayscale images.

4.5 Positioning WAM End-Effector

We continuously update the goal position of the WAM end-effector until the generalized Cartesian space error (including position and orientation)

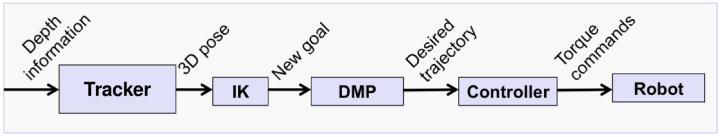

Figure 4.2: General scheme of the experiment. The depth information from the Kinect camera is used to obtain a 3D pose and is sent to the IK algorithm, which computes a new goal. This goal is sent to the DMP, which immediately updates the trajectory. Finally, the desired trajectory to the current goal is sent to the controller.

$$e = \left\| \begin{bmatrix} e_p \\ e_0 \end{bmatrix} \right\|_2 + \delta \tag{4.4}$$

is below a threshold. Here e_p, e_o are the position and orientation errors with respect to the desired pose of the target object, as defined in Section 3.7 of (Siciliano et al., 2009), and δ is a small offset defined as half the length of the bounding box.

The position of the target is defined as the centroid of the points in 3D space, enclosed within the bounding box. In order to determine the orientation, we fit a plane to the 3D points, corresponding to three corners of the bounding box and compute its angular displacement relative to the camera axes. Once the error e is less than a threshold, the gripper closes its fingers and grasps the object. Using a robust Inverse Kinematics (IK) algorithm (Colomé and Torras, 2012, 2015), we can convert the generalized position of the object into a joint vector goal that places the robot in the target.

Then, the robot's goal can be updated online by using Dynamic Motor Primitives (DMP) (Ijspeert et al., 2013) at a joint level, where a desired trajectory of the robot is computed with a second order dynamic system:

$$\dot{\mathbf{z}}/\tau = \alpha_z \left(\beta_z \left(\mathbf{G} - \mathbf{q}\right) - \mathbf{z}\right) + \mathbf{f}(t), \tag{4.5}$$

where $\mathbf{q}$ is the joint position, $\mathbf{G}$ the goal position, α_z, β_z are proportional-derivative constants, τ a time constant, $z = \dot{\mathbf{q}}/\tau$ a rescaled velocity, and $\mathbf{f}(t)$ a shaping function of the trajectory.

This characterization gives us a desired acceleration, velocity and position at each time instant. For $\beta_z = \alpha_z/4$, the system is critically damped and these derived signals can be sent to any controller to track the desired trajectory. In the case of object tracking, the shaping function $\mathbf{f}(t)$ can be set to zero to have a pure critically damped attractor to the goal $\mathbf{q} = \mathbf{G}$, or used as an obstacle-avoidance term as in (Ijspeert et al., 2013).

The DMP representation allows us to change the trajectory goal online, while maintaining the continuity on the position and velocity commands, and without needing to recompute the whole trajectory (as we would have to if using splines). Thus we can update the goal according to the movement of the tracked object using an adequate inverse kinematics algorithm and the default PID controller provided with the arm. A general scheme of the whole tracking system implemented can be seen in Fig 4.2.

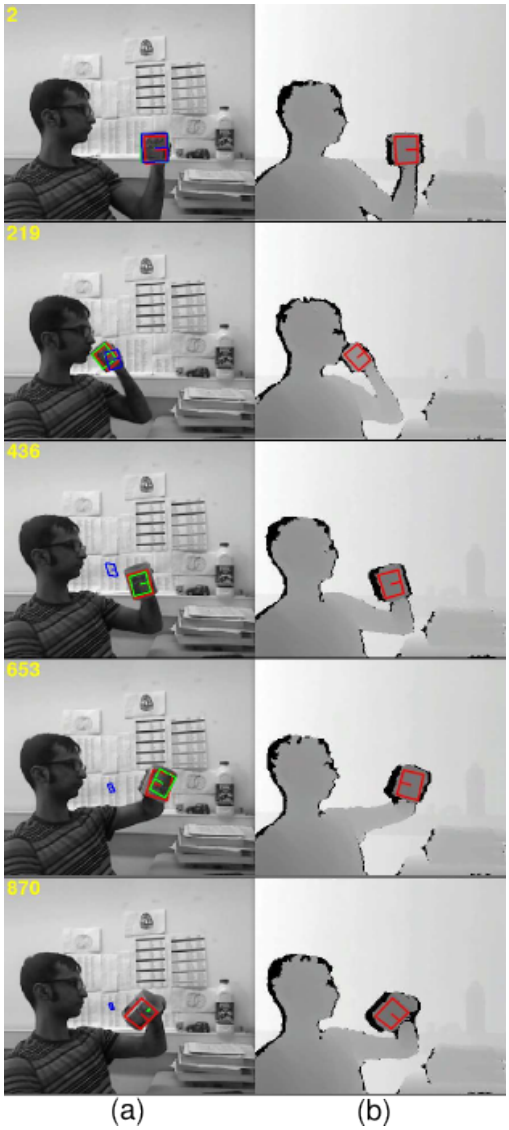

Figure 4.3: Comparison of different tracking algorithms for grayscale (a) and range images (b). Our tracker (red rectangle) uses range data only, whereas the tracker from (Kwon et al., 2009) (blue rectangle) and the one from (Ross et al., 2008) (green rectangle) use grayscale images.

4.6 Results

4.6.1 Pros and Cons of Using Depth Instead of Color Images

Depth data lacks texture information which could be essential to constrain the affine transform. However, we found that texture can also be a limiting factor during tracking, as can be seen when looking at the example shown in Fig. 4.3. During manipulation, the texture on the cup changes rapidly in the image because of its changing orientation

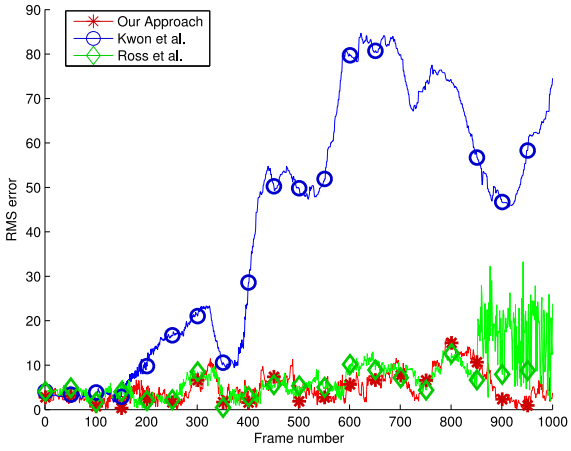

Figure 4.4: A comparison of RMS error of the centroid with respect to the ground truth for our tacker (red color), the tracker of (Kwon et al., 2009) (blue color) and the tracker of (Ross et al., 2008) (green color).

and self-occlusions, while the surface shape remains approximately the same, allowing our tracker to succeed in cases where others fail. This behavior has been observed in several cases. We also tested texture-based trackers for gray-scale images directly on range images, not surprisingly they failed as well.

Figure 4.4 shows a quantitative comparison using the root mean square (RMS) error of the centroid of the bounding box with respect to the ground truth at each time instant for the video corresponding to Fig. 4.3. To generate the closest possible ground truth, we first over-segmented the video using the method proposed in (Grundmann et al., 2010), and then manually relabeled the segments that belong to the tracked object.

It can be seen in Fig. 4.4 that the tracker from (Kwon et al., 2009) and the one from (Ross et al., 2008) eventually loose track ($\sim$ frame 169 and $\sim$ frame 853, respectively), whereas ours successfully tracks the object until the end.

4.6.2 Comparison with the OpenNI Tracker

We compare our method with the OpenNI tracker for the Kinect sensor in the Point Cloud Library. The OpenNI tracker directly determines the 3D rigid transform from the points on the tracked surface model and predicts the six rigid transformation parameters using a particle filter.

In order to make a fair comparison, we omitted the color information in the measurement function, yielding

$$h(p,q) = \sum_j \left(1 + |p_j - q_j|^2\right), \tag{4.6}$$

where j ranges over to all the points in the reference model, p_j is the 3D position of the predicted point, and q_j is the 3D position of the nearest point in the input point cloud.

We conducted several experiments and observed that the OpenNI tracker is more prone to failure due to occlusions than ours, presumably because it does not employ a

Figure 4.5: Comparison of ((a) and (c)) our tracker (red rectangle) with ((b) and (d)) the tracker available in the Point Cloud Library (blue pixels). The color images are shown here for illustration only and not used in the tracking procedure.

motion model. Figure 4.5 shows two of the cases where the OpenNI tracker failed to track the object while ours kept on tracking.

4.6.3 Tracking and Grasping of Moving Objects

Several experiments were conducted with objects of different shapes and appearances, e.g., a milk bottle, a carton box, and a ball. In all experiments, the WAM arm was able

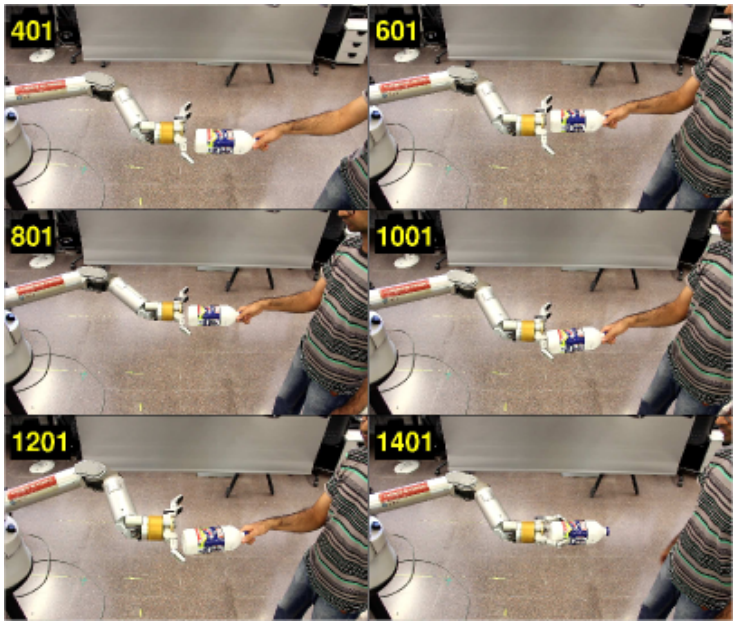

Figure 4.6: Tracking and grasping of a moving bottle.

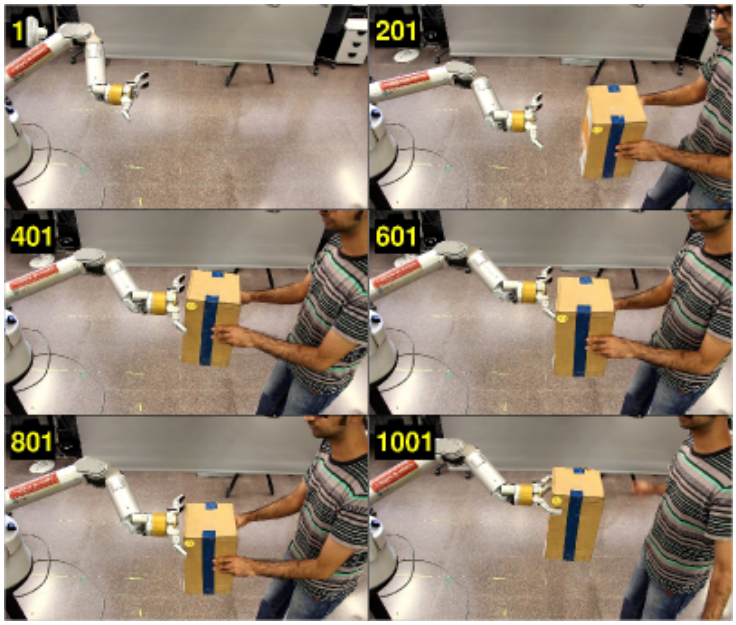

Figure 4.7: Tracking and grasping of a moving carton box.

to successfully chase the object using the tracking information and eventually grasp it. Selected frames are shown in Fig. 4.6 and Fig. 4.7.

Figure 4.6 shows a bottle that is being manipulated by a human. This scene was recorded with a Kinect camera mounted above (the same as illustrated in Fig. 4.1).

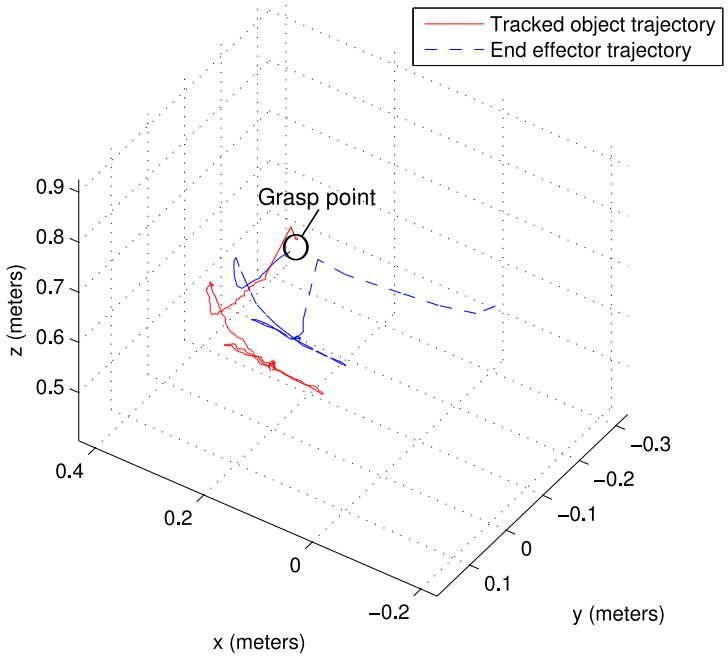

Figure 4.8: Object and end-effector trajectories during tracking and grasping of a moving box.

During the experiment, the robot arm follows the motion of the bottle and tries to minimize the distance between its gripper and the bottle (see frames 401-1201). Once the distance is below a threshold, the gripper closes and grasps the object (see frame no. 1401). Similar results were obtained for other objects, see Fig. 4.7. Figure 4.8 shows the trajectory of a box and the end-effector of the manipulator during one of the experiments.

4.6.4 Tracking-Speed Analysis

The tracker is implemented in C++ and runs at $\sim$ 20 fps on an Intel Xeon quad core processor. We expect further speedup by porting the code to a GPU. The maximum object speed with which our tracker can cope with depends on factors such as the size of the object in the image plane, its shape, and also the way it is being manipulated. In order to get an idea of the speed that our tracker can handle during the experiments, we tracked a cylindrical surface which moved freely in 3D space at a speed that allowed correct tracking of the object (see Fig. 4.9(a)). Figure 4.9(b) shows the distance that the object covered within each second. The tracker was able to track objects moving up to 21 cms/sec.

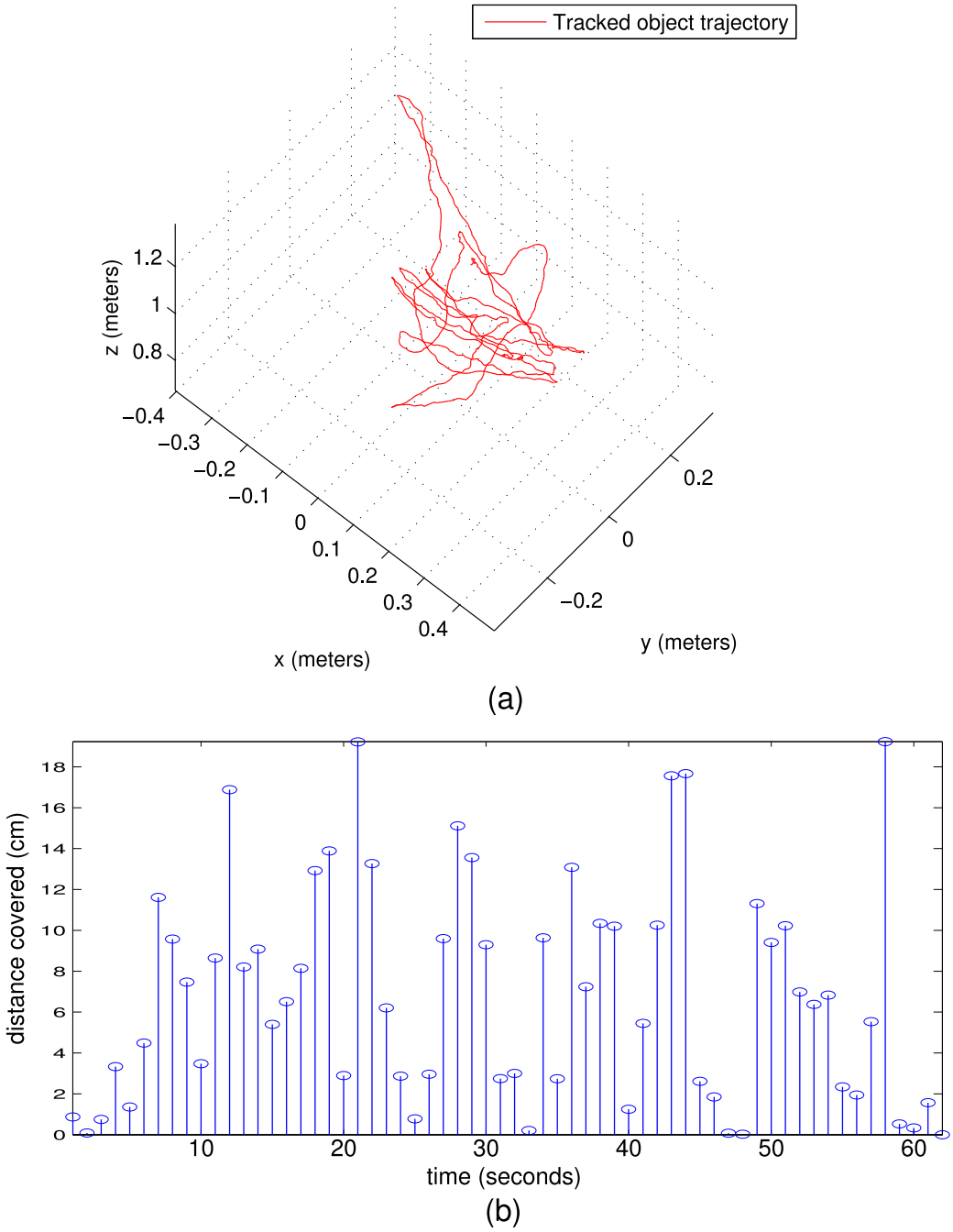

Figure 4.9: Trajectory of tracking a cylindrical surface manipulated freely in 3D space (a) and the distance it covered within each second (b).

4.7 Summary and Future Work

In this chapter we presented a complete system which is able to robustly track and grasp moving objects in 3D space. Our code and complete videos are available for download

at http://www.iri.upc.edu/groups/perception/#trackGrasp.

We demonstrated that reliable realtime tracking can be achieved and used for robotic manipulation of moving objects using depth data alone. For this purpose, we developed a novel method for tracking in depth images based on a geometric particle filter on the affine group. This type of tracking paradigm has been used before in color images (Kwon et al., 2009; Kwon and Park, 2010). An advantage of our method compared to color-based tracking is that its performance is independent of the appearance of the object in terms of color and texture (see Fig. 4.3). Compared to the OpenNI tracker, our method showed equal performance in most cases and even outperformed it in some cases (see Fig. 4.5).

The system has some limitations which we plan to address in the future. If the manipulator moves to a location where it partially occludes the tracked object or if the object undergoes self-occlusions, then the centroid may deviate from the correct position, and the gripper may be moved away from the desired grasping position. This limits the workspace of the manipulator. In the future we plan to use a setup with multiple range sensors to cope with this situation.

In situations where a user supplies a bounding box which does not contain the entire surface to be tracked, our tracker may fail. For instance, we cannot track a small planar patch inside a plane. For the aforementioned reason, our tracker requires that the four corners of the bounding box supplied initially by the user lie on the edges of the surface that is to be tracked.

Chapter 5

Segmentation-Aware Tracking

The local boundaries in a depth image can disappear when different surfaces come in physical contact. This can confuse a tracker that relies on well-defined depth discontinuities. Video segmentation can be used to assist the tracker and help reducing these deficiencies.

In this chapter we proceed a step further towards the robustness in the tracking approach that was introduced in Chapter 4. Robustness is achieved by using mechanisms borrowed from the segmentation scheme introduced in Chapter 3, and integrating them with our tracker from Chapter 4.

5.1 Introduction

Surface tracking in the domain of range image sequences has two main components: (i) extract the surfaces and establish the correspondence of the surfaces over the frames in the sequence of range images, and (ii) compute the motion transformation using these surface correspondences (Sabata and Aggarwal, 1996). Both tasks are intertwined, as the correct extraction of surface patches helps finding correct correspondences, and vice versa. As long as surfaces are spatially disconnected, problem (i) can be more or less easily solved by clustering the 3D points based on their spatial proximity (Dellen et al., 2013; Husain et al., 2015; Jiang et al., 1999). Problem (ii) can be solved by assuming 3D rigid-body motions between extracted point sets (Sabata and Aggarwal, 1996; Kim and Kim, 2010; Sandhu et al., 2010). However, as soon as surfaces get in physical contact with each other, the problem becomes far more challenging, because in this case it is often impossible to distinguish between different objects based on depth differences alone. The situation becomes even more severe when both the manipulator and the manipulated surface undergo the same transformation at this time, e.g., during a hand-object manipulation. In this case, (i) and (ii) need to be solved conjointly, while taking the motion history of the objects into account.

This chapter offers a solution to this problem by combining our segmentation approach as described in Chapter 3 with our particle-filter-based tracker from Chapter 4 with some modifications. Because we use a region growing procedure which automat-

ically adapts to the dynamic range data, predictions from the particle filter can be incorporated in a straightforward manner by refining seeding, and, in consequence, the input to the tracker.

5.2 Related Work

Tracking has previously been performed mostly for color/gray-scale image sequences (Yilmaz et al., 2006; Kwon et al., 2009; Babenko et al., 2011; Fan et al., 2005; Chen et al., 2013). However, depth images pose different challenges to the tracking algorithm than color/gray-scale images.

Most methods for tracking in range images use *a priori* knowledge of 3D point correspondences and find the affine or rigid-body transformation on this basis (Huang and Netravali, 1994). These methods mainly work for sparse data sets, but are less useful when working with dense range images. Other techniques match surface patches instead, eliminating the need for finding exact point correspondences (Sabata and Aggarwal, 1996). The range data is segmented into surface patches, then correspondences are established between patches of adjacent frames, and the motion transformation is estimated. However, such an approach is only effective if the initial (presumably correct) segmentation can be maintained over time. This is however not a trivial task, as small variations in the data and motions can change the segmentation drastically. To overcome this problem, a seeding and region growing technique for range-image sequences was proposed in (Dellen et al., 2013; Husain et al., 2015; Jiang et al., 1999). Maintenance can be improved this way, but when two or more surfaces are in physical contact with each other, it remains difficult to determine the boundary between the surfaces in contact using depth differences alone (Dellen et al., 2013; Husain et al., 2015).

To cope with the specific characteristics of range data, some existing approaches put limitations on the tracked surface by considering only articulated motion (Ganapathi et al., 2010; Knoop et al., 2006; Tsap and Shin, 2004). This simplifies the tracking problem but also limits the usability of the algorithm to particular scenarios. Robust tracking of human hands assuming articulated motion constrained by the 54-dimensional parameter space has been performed in (Oikonomidis et al., 2012). In (Krainin et al., 2011), object tracking using a depth camera was performed (for 3D object reconstruction), but here the robotic hand had to be separated from the range data before applying the algorithm.

The Iterative Closest Point (ICP) algorithm has also been employed for tracking. However, the basic ICP method (Besl and McKay, 1992) is a pairwise matching algorithm which does not take into account past measurements (Rusinkiewicz and Levoy, 2001; Sandhu et al., 2010), hence the error starts to propagate. The ICP algorithm has been previously combined with Kalman filtering for object reconstruction (Foix et al., 2010). However, in this case, the background was removed, leaving only the target object. In cluttered scenes, this approach may thus not be applicable. Point-to-point matching in 3D space requires a high accuracy in the estimation of the transformation matrix. This makes these approaches less suitable for our scenario because of the limited accuracy of the depth camera. Several variants of ICP which achieve better point set registration have been proposed, such as the expectation-maximization ICP (Granger and Pennec, 2002) and softassign (Gold et al., 1998). However, these variants have a

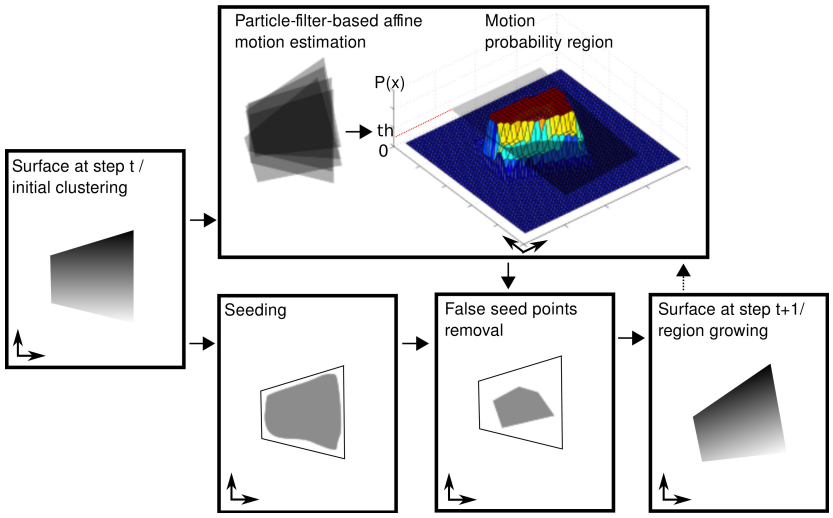

Figure 5.1: Schematic illustrating the basic idea of our approach (for further explanations, see main text).

higher computational cost and require specialized hardware such as GPUs to achieve real-time performance (Tamaki et al., 2010).

In this chapter, we combine seeding and growing of surfaces with particle filtering to overcome the aforementioned limitations. Our main contribution is a robust mechanism for identifying a set of points belonging to a target object that is being manipulated in 3-D space, regardless of its physical contact with other objects.

5.3 Method

Our tracker requires a range image as input at each time step. The algorithm for surface tracking consists of the following steps. Initially a set of non-overlapping geometric surface patches are obtained by segmentation and the surface that we want to track is identified manually (see Section 5.3.1). Segmented surfaces at step t are used to create seed regions in the next frame $t+1$ by comparing the predicted depth values (from quadratic surface models fitted to the segments) to the actual depth values (see Section 5.3.3). At the same time, a motion-probability region is found by the particle-filter-based tracker through the prediction of future states (see Section 5.3.2). The extracted motion-probability region is used to refine the seeding. Region growing allows reconstructing the segment at $t+1$. Based on this segmented area, the translation parameters of resampled states are re-estimated (see Section 5.3.4), which, provided the segmentation is correct, improves the predictions of the tracker for the next frame. The basic idea behind our approach is illustrated in Fig. 5.1.

The adaptive surface fitting for segmentation of 3-D points is used for both initial segmentation, seeding and region growing during the tracking. At each time step t, a depth image $F_t(u,v)$ is acquired by the depth sensor, where (u,v) are the pixel positions in the image grid of size (image length $\times$ image width $\times$ 3), containing the 3D data $x(u,v)$, $y(u,v)$, and $z(u,v)$.

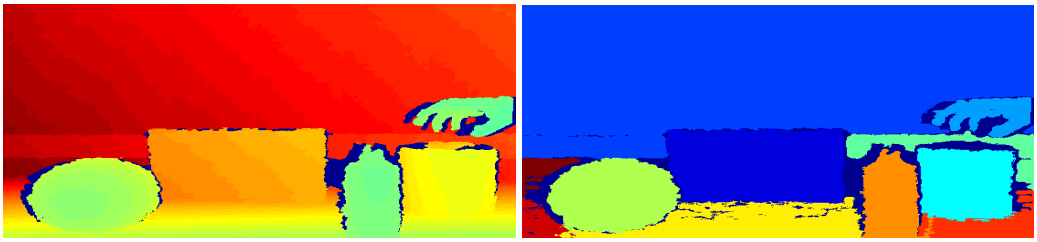

Figure 5.2: (a) Color-coded depth image (Kinect) and (b) segmentation result. Each segment has a unique color.

5.3.1 Initial Segmentation

The segmentation of the first frame is obtained by iterating over the algorithm described in Section 3.3 over a single range image. Fig. 5.2 shows a typical segmentation result obtained with the segmentation. We manually identify the label of the surface that we want to track. We represent the points on the tracked surface as the 2-vector T which is a subset of the image grid.

5.3.2 Particle-Filter-Based Affine Motion Estimation

In order to determine the location of the projected tracked surface, we use the particle filter based tracker as described in Section 4.4. Since we know the actual shape of the template that we want to track (from the initial segmentation), we can track the points that are inside the segment silhouette instead of using a bounding box as we did in Section 4.4.

We generate a motion-probability-region $P_t(u,v)$ based on the re-sampled particles as $P_t(u,v) = 1$ if $\Omega_t(u,v) < \tau$, and 0 otherwise, with $\tau = q/3$, with q being the number of particles and $\Omega_t(k,l) = \sum_{i=0}^{q-1} W_t^i(k,l)$. W_t^i is a binary image which represents the tracked object-surface silhouette determined from the affine state matrix X_t^i (see Eq. 4.1) of the re-sampled particle i, i.e.,

$$W_t^i(k,l) \;=\; \begin{cases} 1 & \text{if} \quad (k,l,1) \in \mathit{\Pi} \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where $\mathit{\Pi} = \left\{ X_t^i \times (u,v,1) \mid \text{for all } (u,v) \in T \right\}$. The motion-probability-region is defined in order to determine the most probable location of the seed for the tracked surface. We achieve this by considering the output of all the resampled particles, i.e., summing over all the resampled particles W^i, and applying a threshold afterwards.

5.3.3 Seeding and Region-Growing Procedure

In order to generate seeds, we first estimate the surface models for each segment. Based on these, we compute the difference between the predicted depth and the actual depth (see Eq. 3.10), providing a seed area for each segment as done in Section 3.9-A1.

For the target segment s_i (that is being tracked) we refine the seed area by removing all points that are not inside $P_t(u,v)$, and prohibit all other segments $j \neq i$ to have

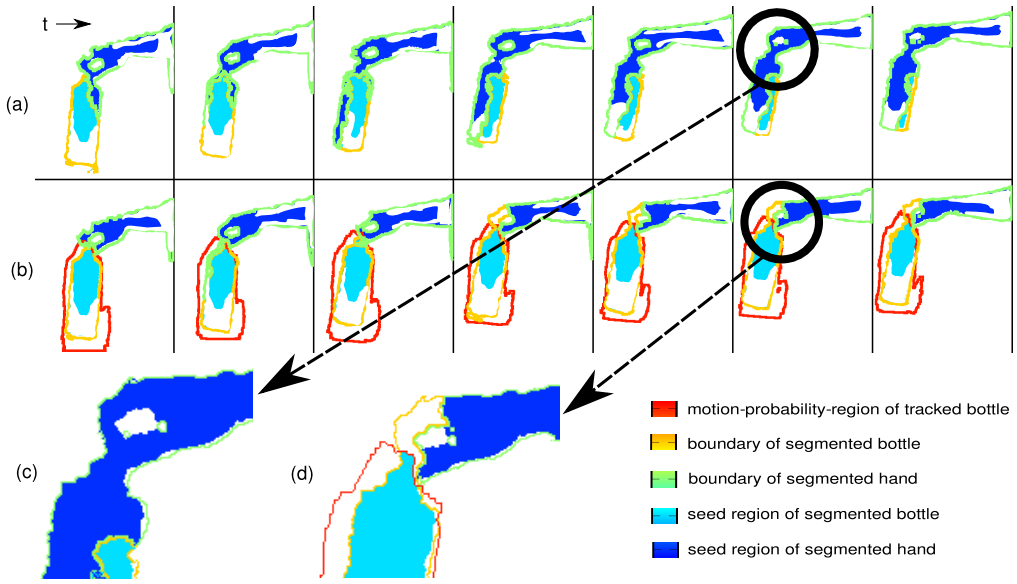

Figure 5.3: Illustration of the improved seeding procedure for a set of frames at regular intervals of time. (a) Seeds without refinement, (b) seeds with refinement, (c) enlarged inset of Fig. 5.3(a) and (d) enlarged inset of Fig. 5.3(b).

seeds inside the probability region of the target, yielding

$$m'_t(u,v) = \begin{cases} 0 & \text{if } P_t(u,v) = 0 \text{ and } s_t(u,v) = i \\ 0 & \text{if } P_t(u,v) = 1 \text{ and } s_t(u,v) = j \\ m_t(u,v) & \text{otherwise.} \end{cases} \tag{5.2}$$

In Fig. 5.3, the basic idea behind the method is illustrated for a scene showing a hand manipulating a bottle. Figure 5.3(a) shows the seeding procedure without using the refinement step. It can be observed that with passage of time, seed points of the hand aggregate in the region of the bottle. This results in error propagation and after every time instant the segmentation/tracking result gets worse. The reason is that the region growing approach is a least-squares minimization, which does not take into account factors such as shape deformation. Because of the similarity of the adjacent regions of the bottle and the hand, and the limited resolution of the data, the hand and the target object get merged. Using the motion-probability-region in the seeding process prevents the bottle and the hand to be merged (see Fig. 5.3(b)).

Once we have obtained the seeds for all the surfaces, we determine the label for all the unlabeled points using the region growing procedure as in Section 3.9-A3.

5.3.4 Refining Re-sampled Particles

In case of depth data, the employed quadratic surface fitting provides segmentation of points whose spatial arrangement varies smoothly in 3-D space and is invariant to illumination changes. For the aforementioned reason, we did not find any advantage of periodically updating the target appearance model (Jepson et al., 2003; Kwon et al., 2009) and hence omitted this step. A simple computation of the mean of the tracked target proved to be sufficient to filter quantization noise in depth data.

We refine the resampled particles of the tracker by averaging the translation component of the affine state matrices $X_{0:q-1}$ from the current time step according to

$$X'_{t,0:q-1} = \begin{bmatrix} A_{t,0:q-1} & (k_{t,0:q-1} + \mathfrak{c}_t)/2 \\ 0 & 1 \end{bmatrix}, \tag{5.3}$$

where $\mathfrak{c}_t = (\overline{x}_t, \overline{y}_t)$ is the centroid of the adaptive segment, obtained from the seeding and region growing procedure, which feeds back to the particle filter. The refined states $X'_{t,0:q-1}$ are used by the tracker for prediction at time step $t+1$.

5.4 Results

We tested the tracking algorithm on several in-house recorded depth videos of hand-object manipulations using a Microsoft Kinect and a PMD camera. The results for selected frames are shown in Figs. 5.4, 5.5 and 5.9. All the datasets along with the tracking results are made available for the readers [1]. In each of the examples, our goal is to track the manipulated object. To the best of our knowledge, to date there is no benchmark dataset for depth videos publicly available. Results obtained with our method are shown for selected movie frames.

5.4.1 Tracking Under Large Translations and Rotations

Figure 5.4(a) shows a human hand moving a cup such that it undergoes large changes in orientation and position. From frame 104 to frame 188, both the hand and the cup go through the same transformation. Nevertheless, the method succeeded to track the cup correctly, even though the two surfaces got in smooth continuity. Figure 5.4(b) shows example of a human hand displacing a bottle from one spot to another. From frame 65 to frame 93, the bottle was tracked correctly. During this time, it was at the same depth and in physical contact with the surface on top of which it was placed. Figure 5.4(c) shows a human manipulating another bottle. This time we tested if our tracker is able to handle a full rotation of 180 degrees involving also translation and scaling. Despite these challenges, the shape of the bottle could be preserved while being tracked. Figure 5.5 shows tracking result for a cylindrical object with a low resolution PMD camera (200×200 pixels). The object was lifted up from the ground and then manipulated in different ways. It can be seen that the object was tracked correctly before, during, and after the human hand manipulated it.

5.4.2 Performance Evaluation and Comparisons

We calculate the RMS error e_{rms} (in number of pixels) between the ground truth centroid $(\overline{x}, \overline{y})$ and the estimated centroid $(\overline{x}', \overline{y}')$ location for quantitative analysis according to

$$e_{\text{rms}} = \sqrt{(\overline{x} - \overline{x}')^2 + (\overline{y} - \overline{y}')^2}. \tag{5.4}$$

Figure 5.6 shows a comparison of the RMS error at each time instant for the cup sequence (Fig. 5.4(a)). The range image sequence was used to track the cup with our approach, i.e., surface fitting along with refining seeds (see Eq. 5.2) and the video

[1] http://www.iri.upc.edu/people/shusain/tracking_data.html

Figure 5.4: Tracking results (red color) for a hand (a) manipulating a cup, (b) moving a bottle and (c) manipulating another bottle using our proposed approach. Depth images are shown using grayscale gradients from black (near) to white (far). For illustration, color image has been overlaid in frame 006, frame 02 and frame 001 of Fig. 5.4(a), Fig. 5.4(b) and Fig. 5.4(c) respectively. The data were recorded using a Microsoft Kinect camera.

segmentation from Chapter 3. After frame 80, the surface of the cup got in smooth continuity with the hand and henceforth the surface fitting procedure alone was unable to disambiguate the boundary between the two surfaces and lost tracking, whereas when it was combined with the particle filter, the cup was successfully tracked.

Figure 5.6 also shows the results using two different particle-filter-based trackers from (Kwon et al., 2009) and (Ross et al., 2008). We used the corresponding color images of the cup sequence as an input for these trackers. It can be observed that these trackers have a higher RMS error when compared to ours. This is because the accuracy of color-based trackers depends on the richness of the texture in the color image.

We also compute the difference e_{size} between the size (in number of pixels) of the tracked surface in the image plane s' and the size of the surface in the ground truth s, i.e.,

$$e_{\text{size}} = |s - s'|. \tag{5.5}$$

Figure 5.7 shows a comparison of the region size using the four approaches described

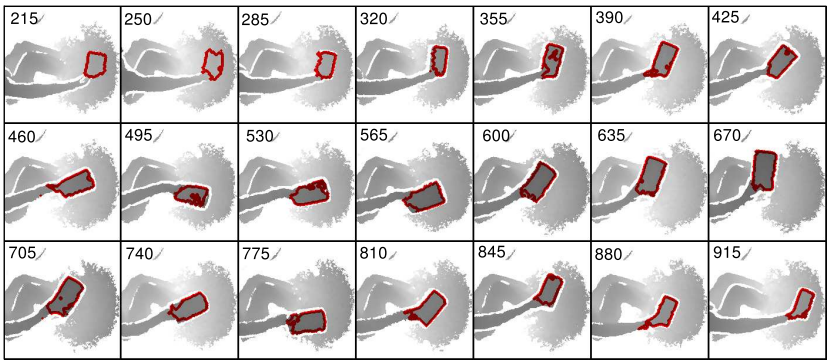

Figure 5.5: Tracking results (red color) for a cylindrical object manipulated by a human hand, using our proposed approach. The data was recorded using a PMD camera. Depth images are shown using grayscale gradients from black (near) to white (far).

Table 5.1: Comparison of the average RMS and the region size error for the cup sequence.

	Our Approach	**Dellen et al.**	**Ross et al.**	**Kwon et al.**
$\overline{e}_{\text{rms}}$	5.69	116.86	13.20	76.52
$\overline{e}_{\text{size}}$	542.5	4368.1	2827.3	2983.6

earlier. In Fig. 5.6 and Fig. 5.7 it can be seen that the method from (Ross et al., 2008) (green color), even though it was able to track the surface, could not maintain its size.

Table 5.1 shows the average of the RMS error and the region size error for the four approaches plotted in Fig. 5.6 and Fig. 5.7.

We also compare our approach to an ICP-based tracker (Rusinkiewicz and Levoy, 2001). We compute the mean of the nearest neighbor distances for each point on the tracked surface with respect to the ground truth. To find the nearest neighbors, we used the approach from (Friedman et al., 1977). Figures 5.8(a) and (b) show a comparison for the sequences shown in Fig. 5.4(a) and (c), respectively. The ICP algorithm was able to keep track of the surfaces but failed to correctly determine surface orientation, hence we see a greater average nearest neighbor distance with respect to the ground truth when the surface is rotating. Our approach clearly outperforms the ICP-based tracker. To create the ground-truth, we first over-segmented the video using the method proposed by (Grundmann et al., 2010) and then manually relabeled the segments that belong to the tracked object.

5.4.3 Increased Particle-Filter Effectiveness

In our approach, the translation component of the affine state matrix is refined by re-computing it after region growing (see Eq. 5.3). Hence, we expect a better performance from the particle-filter-based estimator when the translation component of the tracked object dominates other kinds of transformations. This can be illustrated by tracking a spherical surface, since it can undergo translation and scaling only. Figure 5.9(a) shows

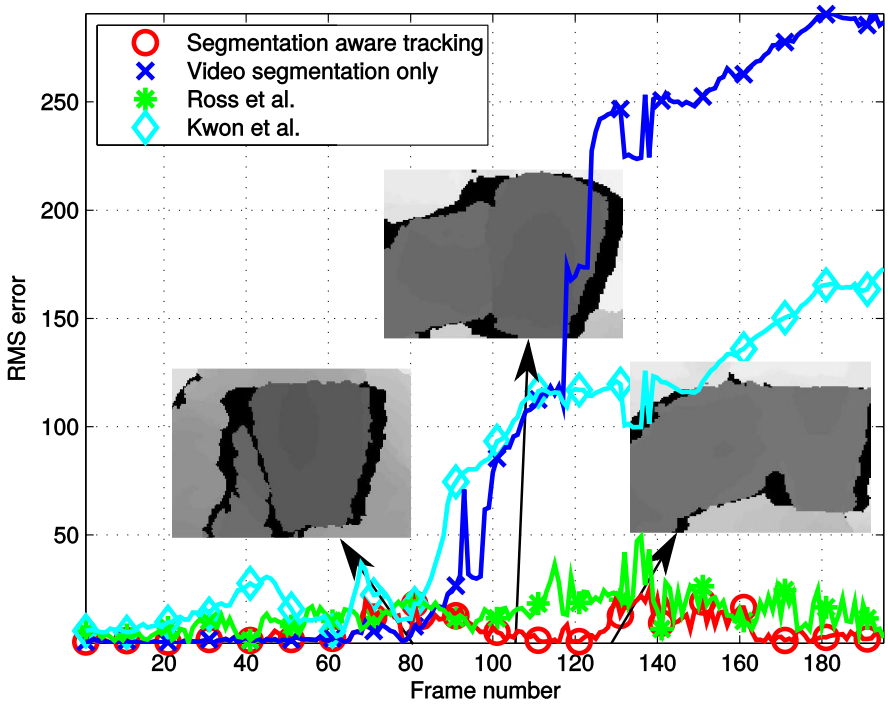

Figure 5.6: Comparison of the RMS error of the centroid with respect to the ground-truth for tracking with our approach (in red), video segmentation only from Chapter 3 (in blue), the tracker from (Ross et al., 2008) (in green), and the tracker from (Kwon et al., 2009) (in cyan) shown for the cup sequence (Fig. 5.4(a)). Enlarged insets of selected frames from the cup sequence are also shown.

selected depth images from (Dellen et al., 2013) together with the tracking results (red color). We determine the efficiency, by finding the number of effective particles, i.e., $N_{\text{eff}} = 1/\sum_i (\tilde{w}_t^i)^2$, as defined in (Doucet et al., 2000; Kwon et al., 2009), where $\tilde{w}^i$ are the normalized importance weights (for details see (Kwon et al., 2009)). The number of effective particles provides a measure of how well the tracker managed to predict the future state of the object. Figure 5.9(b) shows N_{eff} with (blue line) and without (red line) recomputing translation. We have used 15 particles in all our experiments, hence N_{eff} can vary between 1 (worst) to 15 (best). Clearly, the number of effective particles increases after applying the refinement.

5.4.4 Implementation Details

Currently, the algorithm is able to process $\sim$ 2 frames per second for a frame size of 200×200 pixels in Matlab on Intel Xeon 3.3 GHz processor. We have implemented the particle filter in C++ which runs at $\sim$20 frames per second. With a complete C/C++ implementation of the method, we expect to reach real-time performance.

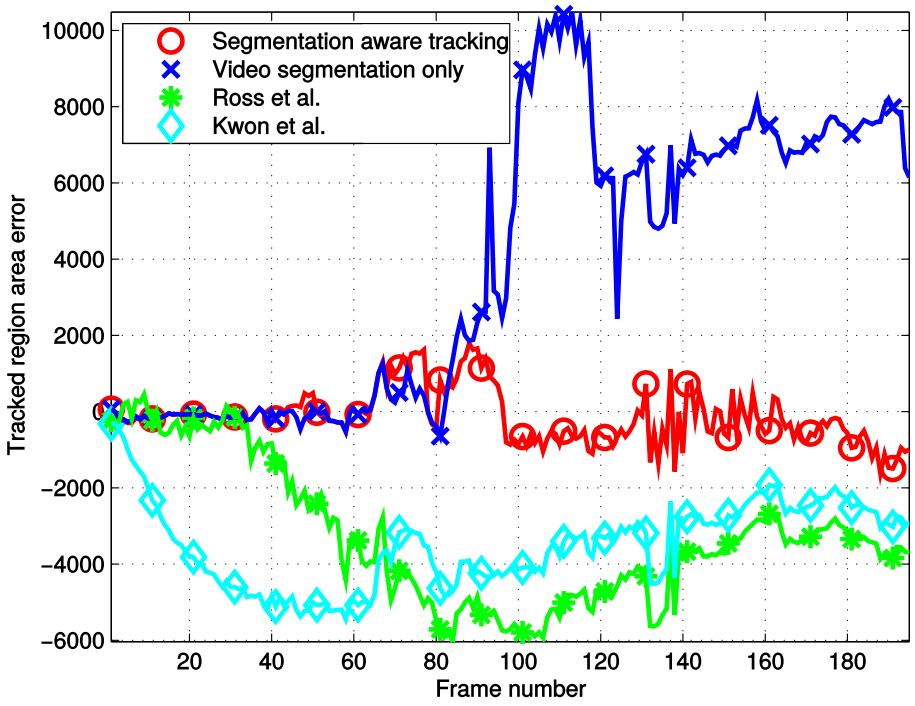

Figure 5.7: Comparison of error in the size of the tracked region with respect to the ground-truth for tracking with our approach (in red), video segmentation only from Chapter 3 (in blue), the tracker from (Ross et al., 2008) (in green), and the tracker from (Kwon et al., 2009) (in cyan) shown for the cup sequence (Fig. 5.4(a)).

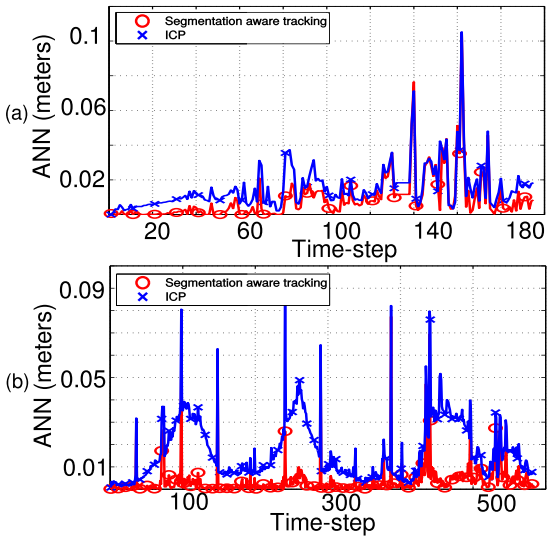

Figure 5.8: Plots of Average Nearest Neighbor distances ((a) and (b)) with respect to the ground truth, for the sequences shown in Figs. 5.4(a) and (c) respectively.

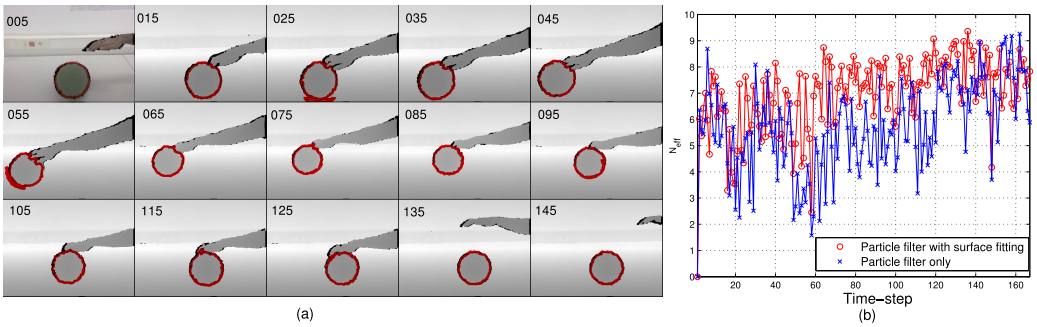

Figure 5.9: (a) Tracking a rolling ball. For illustration, color image has been overlaid in frame 005. (b) Plot of N_{eff} of the rolling ball.

5.5 Summary and Future Work

We presented a novel approach to surface tracking by combining a state-of-the-art particle-filter-based tracker with a clustering method based on surface fitting. The combination of both methods allowed tracking of object surfaces in videos acquired with depth cameras (Kinect and PMD) despite their limited resolution and accuracy. Object surfaces could be tracked correctly even in situations where the object got in contact with other objects or got touched by the manipulator, assimilating the shape of the tracked object, which represents a highly challenging test case for tracking in depth movies. Since our method takes past measurements into account, errors arising in the method can be reduced, leading to an increased performance as compared with an ICP-based tracker, as seen in Figure 5.8.

The method could fail to track a complex surface that cannot be well fitted using a second order polynomial equation. Tracking performance is also dependent on the depth resolution of the range sensor. For example, the boundary of a tracked surface might become totally indistinguishable, when touching other surfaces. In such cases, the color based trackers could be used to compliment our method.

The proposed approach could be extended to track multiple objects simultaneously while maintaining segmentation by employing multiple particle filters to model the individual motion of the different surfaces.

Chapter 6

Object Recognition

Image and video segmentation as introduced in Chapter 3 often serves as the first step towards the development of different high-level vision related tasks. In this chapter we will discuss one such task, i.e., object recognition. The focus is on the exploitation of the natural 3D structure of a segmented scene using a graphical model. In particular, a new supervised learning approach for the recognition of objects from 3D point clouds using Conditional Random Fields is presented. The method allows for learning and inference from unorganized point clouds of arbitrary sizes and shows significant benefit in terms of computational speed during prediction when compared to a state-of-the-art approach based on constrained optimization.

6.1 Introduction

Range sensing devices using active illumination for depth estimation such as Microsoft Kinect and LIDAR provide a 3D representation of the scene in the form of a point cloud which can be used for robot perception. In this context, the semantic labeling of cluttered indoor scenes is highly important for facilitating the interaction of the robot with its environment, for example in mobile robotics or robot manipulation, where objects have to be targeted and grasped. An example of a labeled point cloud according to common object categories for an office environment is shown in Figure 6.1. The goal of semantic labeling is to assign object labels, such as "monitor", "table", or "wall" to the respective parts of the scene. Commonly, the data is first subdivided into parts, then a 3D graphical model is constructed that captures the features and contextual relations of the parts, and used for learning and recognition.

In the past, several methods have been proposed for learning the semantics of objects that exploit contextual information in color images for object recognition (Choi et al., 2012; Quattoni et al., 2004). One problem with color images is that they lack the necessary discriminative properties of the underlying 3D geometry. To overcome this limitation, some approaches extracted first 3D information from the images using stereo (Tombari et al., 2011) or monocular depth cues (Rothganger et al., 2006),

Figure 6.1: An example of a labeled point cloud of a typical office environment.

but due to the limited accuracy of passive methods, this did not lead to a significant improvement, as recently pointed out in (Anand et al., 2013).

Using range data from active sensing for semantic labeling has been shown to improve object recognition (Anand et al., 2013; Lai et al., 2012). Here, many approaches use the local arrangement of individual object parts with respect to each other for object recognition (Savarese and Fei-Fei, 2007; Lai et al., 2012). But besides these local contextual cues describing the object, its global, coordinate invariant geometric relations with other objects provide additional important clues for its recognition (De Laet et al., 2013). Graphical models provide a unified framework that allows incorporating both the object's local and global features and have for this reason been widely used for this task (Anand et al., 2013; Murphy et al., 2004; Li et al., 2011). However, current approaches are rather time-expensive which makes them unfeasible for on-line classification. To reduce the computation times for these models, inherent constraints have to be relaxed leading to a decrease in accuracy.

In this chapter we explore the use of Conditional Random Fields (CRFs) for time-efficient approximate inference. Similar to previous approaches, we use a graphical model to describe the point clouds and encode features and relations of point-cloud segments, representing surface patches. We choose CRFs over the more commonly used Markov Random Fields because it is a more direct approach for modeling the probability of labels (He et al., 2004). This leads to a significant improvement in terms of computation time while yielding predictions with an accuracy similar to the ones obtained with exact inference (Anand et al., 2013). We also propose the use of Point Feature Histogram (PFH) (Rusu et al., 2008b) as a 3D shape descriptor of point cloud segments. This descriptor has recently been shown to outperform others for recognition tasks in terms of precision (Ramisa et al., 2013). We tested the Conditional Random Field framework on challenging RGB-D and range-only datasets. We evaluated our approach on three different kinds of scenes and compared its performance to a state-of-the-art method (Anand et al., 2013).

6.2 Related Work

A large body of work has been conducted in the area of object recognition from 2D images in the past. We can only provide a rough overview here and focus on those methods that use undirected graphs for image representation (Quattoni et al., 2004; Wang et al., 2011; He et al., 2004; Malisiewicz and Efros, 2009; Kumar and Hebert, 2005; Shotton et al., 2006; Rabinovich et al., 2007).

Quattoni et al. (2004) performed object recognition by constructing a graph using different parts of an object together with their relative arrangements. Similar part-based models for object recognition are adopted in (Wang et al., 2011). Using relative arrangements of different objects within a scene for recognition has also been performed in (He et al., 2004; Malisiewicz and Efros, 2009; Kumar and Hebert, 2005; Shotton et al., 2006). In (He et al., 2004; Shotton et al., 2006), regional and global image features are incorporated in a CRF. It is also argued in (He et al., 2004; Quattoni et al., 2004) that when the goal is to estimate only the posterior over the labels given the input data (point cloud in our case), a discriminative model (CRF) is more suitable when compared to a generative one (Markov Random Field). Rabinovich et al. (2007) labeled objects according to their contextual relevance in a post-processing step inside a CRF framework.

Compared to 2D images, far less work has been done for 3D scenes (Munoz et al., 2009a; Lai et al., 2012; Xiong and Huber, 2010; Anand et al., 2013; Husain et al., 2016b). Munoz et al. (2009a) learned Associative Markov Network models using a functional gradient technique for labeling point clouds of outdoor scenes acquired using a LIDAR sensor. Object detection after learning from its multiple isolated views is shown in (Lai et al., 2012). Labeling of planar patches in a point cloud using context is performed in (Xiong and Huber, 2010).

Our work is closely related to Anand et al. (2013), where each segment of the input point cloud is represented by a node in a graph and the relational information between different segments is modeled using pairwise edge potentials. In order to make the model suitable for on-line classification, the constraints in the proposed optimization problem were relaxed which led to a drop in recall. In comparison, our time-efficient approximate inference model yielded results similar to the exact inference as in (Anand et al., 2013).

6.3 Problem Statement

Given a set X of n segments, which is a subset of an over-segmented, input point cloud $\mathcal{X}$, i.e., $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{X}$, the goal is to determine a set of unique semantic labels $Y = \{y_1, y_2, \ldots, y_n\}$ for each segment. Each y_i is a member of the output set $\mathcal{Y}$. Here, $\mathcal{Y}$ is a finite set of k object categories that frequently occur in a particular scene. The segment x_i is essentially a vector of variable length containing information about the position of the sampled points in Euclidean space which can also contain color information.

6.4 Methodology

Our approach begins with the construction of a graphical model of the given point cloud, which is segmented by forming clusters based on the differences in the local surface normals and the connectivity of the surfaces. Each point cloud is represented as a set of nodes and edges, and their potential functions are constructed (Section 6.4.1). During learning (Section 6.4.3), we estimate the node and edge weights that maximize the conditional likelihood of the labels given the input features (Section 6.4.4). We use the same approximate inference method for both the prediction of labels and during learning (Section 6.4.2).

6.4.1 Graphical Model

We use a graphical structure G analogous to (Anand et al., 2013), i.e., each segment x_i in the point cloud is represented by a node which can have exactly one label. An edge exists between two nodes if a distance measure between the corresponding segments is less than a threshold. We define the potential function $\Psi(Y, X; w)$ over unary and pairwise cliques, i.e.,

$$\Psi(Y, X; w) = \sum_{i \in \mathcal{N}} \sum_{l} u_l^{y_i} \cdot \phi_l(x_i) + \sum_{(i,j) \in \mathcal{E}} \sum_{m} v_m^{y_i y_j} \cdot \varphi_m(x_i, x_j), \tag{6.1}$$

where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The parameters $u_l^{y_i}, v_m^{y_i y_j}$ are the components of the parameter vector w. Each node x_i and edge (x_i, x_j) is represented by a discriminative node feature vector $\phi_l \in \mathbb{R}^{d_1}$ and an edge feature vector $\varphi_m \in \mathbb{R}^{d_2}$, respectively. The products $u_l^{y_i} \cdot \phi_l(x_i)$ and $v_m^{y_i y_j} \cdot \varphi_m(x_i, x_j)$ determine the discriminative strength of each node feature ϕ_l for the label y_i and edge feature φ_m for labels (y_i, y_j), respectively.

Unlike (Anand et al., 2013), where the parameter w is optimized for the joint probability distribution $P(X, Y; w)$, we optimize w using the maximum likelihood for the conditional distribution $P(Y|X; w)$ which we define as

$$P(Y|X, w) = \frac{e^{\Psi(Y,X;w)}}{\sum_{y \in \mathcal{Y}} e^{\Psi(Y,X;w)}}, \tag{6.2}$$

The parameter w for optimizing the conditional likelihood (Equation 6.2) is learnt from the training examples which will be explained in Section 6.4.3.

6.4.2 Inference

In order to predict the object category labels, given a set of input segments X and a learned parameter vector w^*, we will choose the labels that give the maximum a posteriori (MAP) labeling of the conditional distribution, i.e.,

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}^n} P(Y|X; w^*). \tag{6.3}$$

Our model assumes a large number of object classes that considering all labels, leads to a computational complexity of $\mathrm{O}(\mathcal{Y}^n)$ which makes exact inference intractable. Hence we

use loopy belief propagation (LBP) (Weiss, 2001; Yedidia et al., 2001) for approximate inference which has been shown to approximate the log-likelihood better than other methods, e.g., mean field approximation (Weiss, 2001).

6.4.3 Learning

For learning the parameter vector w, we use the L2-regularized, conditional log-likelihood as the loss function (Tsuruoka et al., 2009), i.e.,

$$\mathcal{L}(w) = \lambda||w||^2 - \sum_{i=1}^{p} \log P(Y_i|X_i; w), \tag{6.4}$$

where p is the number of training examples and λ is the regularization parameter that prevents over-fitting the parameter w on the training data. To estimate the parameter vector w, i.e., $w^* = \arg\min_w \mathcal{L}(w)$, we use mini-batch stochastic gradient descent (SGD) (Bottou, 1998). In mini-batch SGD, the parameter vector w is updated after every iteration j according to

$$w_{j+1} = w_j - \frac{\eta_j}{b} \sum_{i=1}^{b} \frac{\partial}{\partial w} \mathcal{L}_i(w_j), \tag{6.5}$$

where b is the batch size that we determine empirically by cross-validation on the training data. We use the decreasing step size η_j at each iteration j as proposed in (Tsuruoka et al., 2009), i.e.,

$$\eta_j = \frac{\eta_0}{1 + j/p}, \tag{6.6}$$

where η_0 is a constant. In order to solve Equation 6.5, we need to determine the gradient of the loss function at each iteration j, i.e.,

$$\frac{\partial}{\partial w} \mathcal{L}_i(w) = 2\lambda w - \frac{\partial}{\partial w} \log \left(\frac{e^{\Psi(Y_i, X_i; w)}}{\sum_{y \in \mathcal{Y}} e^{\Psi(Y_i, X_i; w)}} \right) \tag{6.7}$$

We first differentiate Equation 6.7 with respect to the parameters u_l, corresponding to the node features (see Equation 6.1), for a training example i, i.e.,

$$\frac{\partial}{\partial u_l} \mathcal{L}_i(w) = 2\lambda u_l + \sum_{j \in \mathcal{N}} \phi_l(x_j) - \sum_{y \in \mathcal{Y}} P(Y|X; w) \sum_{j \in \mathcal{N}} \phi_l(x_j). \tag{6.8}$$

Note that in Equation 6.8 the segment x and label y belong to the i-th training example. The gradient computed is simply the regularization parameter plus the difference of a feature from its expected value. The expected value of the feature, i.e., $\sum_{y \in \mathcal{Y}} P(Y|X; w) \sum_{j \in \mathcal{N}} \phi_l(x_j)$, is calculated using loopy belief propagation. Similarly, we can differentiate Equation 6.7 with respect to the edge parameters v_m as follows

$$\frac{\partial}{\partial v_m} \mathcal{L}_i(w) = 2\lambda v_m + \sum_{(j,k) \in \mathcal{E}} \varphi_m(x_j, x_k) - \sum_{y \in \mathcal{Y}} P(Y|X; w) \sum_{(j,k) \in \mathcal{E}} \varphi_m(x_j, x_k). \tag{6.9}$$

Table 6.1: Node features computed for each segment x_i for data from kinect sensor

Feature	Count
Histogram of HSV color values	14
Average of HSV color values	3
Average of HOG features	31
Linearity	1
Planarity	1
Scatter	1
Vertical component of the normal	1
Vertical and horizontal extent of bounding box	2
Distance from the scene boundary	1

Table 6.2: Edge features computed for segments x_i and x_j for data from kinect sensor.

Feature	Count
Difference of avg HSV color values	3
Coplanarity and Convexity	2
Horizontal and vertical distance between centroids	2
Angle between surface normals	2
Distance between closest points	1
Relative position from camera	1

6.4.4 Features

The choice of node and edge features mainly depends on the nature of the acquired data. We tested our model on data from two different kinds of range sensors, i.e., Microsoft Kinect (short range, for indoor scenes along with color) and LIDAR (long range, for outdoor scenes).

Features for Kinect sensor

Table 6.1 and Table 6.2 provide a list of the node and edge features, respectively, for data from the Kinect sensor. A detailed description of how these features are computed can be found in (Anand et al., 2013).

Features for LIDAR sensor

Table 6.3 provides a list of the node features that we used for data acquired with the LIDAR sensor. The spectral features for linearity, planarity and scatter are the same as the ones used in (Munoz et al., 2009a,b). In addition we also used the Point Feature Histogram (PFH) and the volume of the convex hull V of each segment. PFH is originally computed by determining relations between every pair of points p_i and p_j, and their estimated normals n_i and n_j in a k-neighborhood, where the neighborhood is

Table 6.3: Node features computed for each segment x_i for data from lidar sensor.

Feature	Count
Linearity	1
Planarity	1
Scatter	1
Volume of Convex Hull	1
PFH	27

usually a sphere with a fixed radius. Different from them, we define this neighborhood as all the 3D points belonging to a segment x_i. Afterwards, all points are binned in a 27-dimensional histogram. We use the concatenation of the spectral features of both nodes x_i and x_j as edge features.

6.4.5 Cumulative Binning

All the features are binned using a cumulative binning strategy (Anand et al., 2013), i.e., each feature is represented by $n_b = 10$ binary values. Instead of creating a single node potential for each node feature as in (Anand et al., 2013), we found that the precision increased after grouping two consecutive bins together and treating them as a single node potential. Hence, in total, the number of weights that we learn are $n_{\text{tot}} = (n_{\text{nodeFeatures}} \times (n_b/2) \times n_{\text{states}}) + (n_{\text{edgeFeatures}} \times n_{\text{states}} \times n_{\text{states}})$, where $n_{\text{nodeFeatures}}$ is the number of node features, n_{states} is the number of labels and $n_{\text{edgeFeatures}}$ is the number of edge features.

6.5 Experiments

We tested our model on the publicly available Cornell rgb-d datasets[1] (CRGBD) (Anand et al., 2013) for indoor scenes and the Oakland 3-D Point Cloud dataset[2] (OPCD) (Munoz et al., 2009a) for outdoor scenes. In CRGBD, one dataset consists of home scenes and another of office scenes acquired with the Microsoft Kinect sensor, along with a human annotation. The OPCD dataset contains outdoor scenes acquired with a LIDAR sensor along with a human annotation. We evaluate our model separately on each of these scenes.

6.5.1 Evaluation Measure

For evaluating the prediction accuracy, we use the precision and recall metric commonly found in the pattern recognition literature (Anand et al., 2013; Munoz, 2013). The precision value p_c of the c-th class is defined as $p_c = t_c/i_c$, where t_c is the number of correctly classified segments (true positives) and i_c is the number of segments predicted as the c-th class. The recall value r_c is defined as $r_c = t_c/n_c$, where n_c is the number of ground-truth segments in the c-th class. In order to aggregate the precision and

[1] Available:http://pr.cs.cornell.edu/sceneunderstanding/data/data.php
[2] Available:http://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc/

Table 6.4: A comparison of average micro precision/recall, and average macro precision and recall for home and office scenes from CRGBD (Anand et al., 2013). Data for *svm_node_only* and *svm_mrf_parsimon* taken from table II in (Anand et al., 2013).

	Office Scenes			Home Scenes		
	micro	macro		micro	macro	
Algorithm	P/R	Precision	Recall	P/R	Precision	Recall
svm_node_only (Anand et al., 2013)	77.97	69.44	66.23	56.50	37.18	34.73
crf_node_only	78.11	69.72	63.45	58.64	39.40	36.41
svm_mrf_parsimon (Anand et al., 2013)	84.06	80.52	72.64	73.38	56.81	54.80
crf_lbp	83.28	79.80	73.90	69.11	54.91	50.82

recall over all the k object categories, we calculate both the micro and macro-average precision/recall (Anand et al., 2013). The micro average gives an average of the precision/recall over the number of samples, whereas the macro average is the average of the individual precision/recall of each category. Hence, in the case of micro average, categories having more samples are given more importance.

6.5.2 Results for CRGBD

We first conducted experiments on the home and office scenes from CRGBD. A graphical model is constructed including a set of nodes and edges for each example. Each segment within the point cloud is used as a node and an edge is drawn between two nodes if the minimum distance between the corresponding segments is less than *context_range* as defined in (Anand et al., 2013). In order to compare our CRF model to the MRF model of (Anand et al., 2013), we used the same node and edge potentials (associative and non-associative) as in (Anand et al., 2013). The examples are divided into training and test sets according to the 4-fold cross validation. We learn the weights using the training set (see Section 6.4.3) and then predict the labels for the test set using the learned model (see Section 6.4.2).

Table 6.4 shows a comparison of our results with the ones obtained in (Anand et al., 2013), using the micro and macro-average precision/recall for CRGBD. Here, *crf_node_only* refers to our model without taking the contextual relations into account which can be interpreted as a multinomial logistic-regression model showing the effect of node features only. The term *crf_lbp* refers to the full model with both node and edge potentials. We obtained very similar results to the multi-class SVM (*svm_node_only*) (Anand et al., 2013). A small drop in precision is observed when comparing approximate inference under the CRF framework (*crf_lbp*) with exact inference under the MRF one (*svm_mrf_parsimon*) (Anand et al., 2013).

Our approach leads to a considerable increase in computational speed, i.e., ∼0.014 seconds, on a single core implementation in C++. In (Anand et al., 2013), the computational time for exact inference (∼18 minutes) was reduced to ∼0.05 seconds by relaxing the constraints in the proposed optimization problem, but this led to a significant drop in recall. In comparison, our model permits time-efficient approximate inference with a smaller decrease in precision and recall when tested on the same dataset along with the same potentials as in (Anand et al., 2013). This makes our model more suitable for on-line classification.

Figure 6.2 and 6.3 show the normalized confusion matrices for the office and home

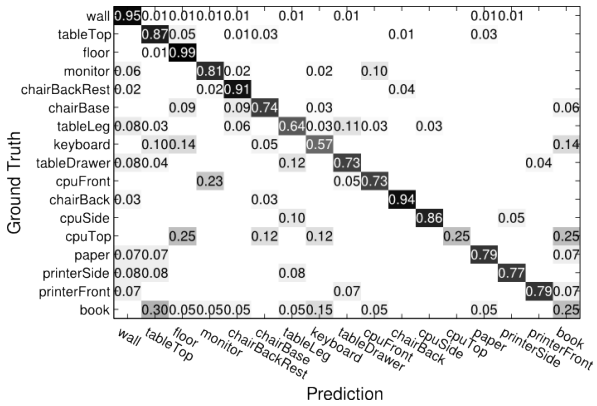

Figure 6.2: Confusion Matrix of our results for the office dataset from CRGBD (Anand et al., 2013) with 17 categories.

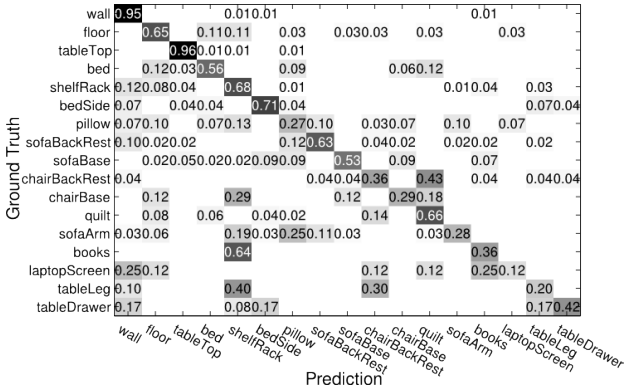

Figure 6.3: Confusion Matrix of our results for the home dataset from CRGBD (Anand et al., 2013) with 17 categories.

datasets from CRGBD, respectively. Our model provides correct predictions for most object categories as indicated by the large values on the diagonals compared to non-diagonal entries of the matrices. In the office dataset, some categories are more easily confused than others, for example, the book lying on a table can get confused with the tableTop (Fig. 6.2). Similar problems are observed in the home dataset, where the books can get confused with the shelfRack (Fig. 6.3).

We made similar observation about the confusion matrices as reported in (Anand et al., 2013), i.e., the object categories in the office scenes were less confused when compared to the home scenes. We also noticed that some of the categories such as the book and table drawer are less confused with our approach.

6.5.3 Results for OPCD

Next, we conducted experiments on scenes from OPCD. As before, a graphical model is constructed for each example, where each segment within the point cloud is used as a node. We define an edge between two nodes if the average distance of the corresponding

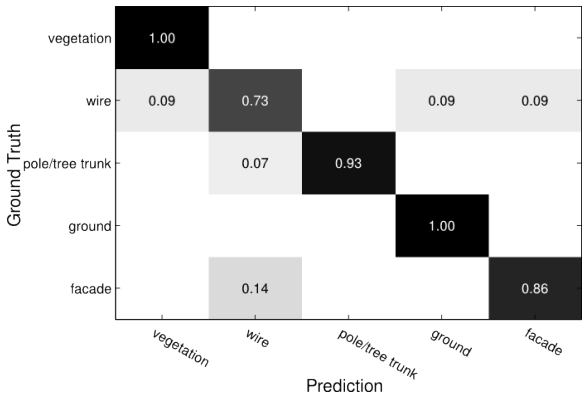

Figure 6.4: Confusion Matrix of our results for OPCD (Munoz et al., 2009a) with 5 categories.

Table 6.5: Average micro precision/recall, and average macro precision and recall for OPCD (Munoz et al., 2009a).

Algorithm	Features	micro	macro	
		P/R	Precision	Recall
crf_node_only	spectral features,V	48.61	46.19	48.22
crf_node_only	spectral features,V, PFH	81.94	80.78	80.04
crf_lbp	spectral features,V	75.00	74.62	74.66
crf_lbp	spectral features,V, PFH	91.67	90.65	90.35

segments is less than 5 m. Here, only the closest nearest neighbors are used, representing 20% of all neighbors. From the 17 examples in OPCD we selected 16 and divided them into training and test sets according to the 4-fold cross validation. We learned the weights using the training set (see Section 6.4.3), and then predicted the labels for the test set (see Section 6.4.2).

Table 6.5 shows the evaluation results for OPCD. Here, *crf_node_only* refers to the model without taking the contextual relations into account, and *crf_lbp* refers to the model with both node and edge potentials. We obtained a considerable increase in precision after using PFH as an additional feature both in *crf_node_only* and *crf_lbp*.

Fig. 6.4 shows the normalized confusion matrix for OPCD. We have less confusion in this scenario. This is mainly due to the smaller number of categories in this dataset.

6.6 Summary and Future Work

We have explored the Conditional Random Field framework for modeling local features and contextual relations of objects from point clouds and tested them on datasets acquired with two different range sensing devices. We chose the mini-batch stochastic gradient descent for optimization, as it is well known for its faster convergence compared to other optimizers such as limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (Vishwanathan et al., 2006). We achieved a similar precision in the prediction

of correct object labels as a state-of-the-art method based on a Markov Random Field framework (Anand et al., 2013) while improving computational speed.

We further showed that the inclusion of the Point Feature Histogram as a feature leads to a significant increase in precision for the OPCD dataset. We did not observe the same effect for the CRGBD. This is because the segments here have a smooth 3D shape for almost all object categories (Anand et al., 2013), as compared to (Munoz et al., 2009a), and thus the Point Feature Histogram does not provide any additional information about the 3D structure of the object.

In the future, we plan to extend our approach to dynamic scenes (Dellen et al., 2013) and incorporate the motion parameters obtained from a tracker (Husain et al., 2014a), during learning. Stochastic gradient descent will allow us to dynamically update the learned parameters for new examples. We also plan to use more advanced inference techniques such as convex belief propagation which guarantees convergence for graphical models with loops (Schwing et al., 2011).

Chapter 7

Object Class Segmentation

In the previous chapters we have used models such as surface fitting, and feature descriptors which were hand-picked, based on the observed characteristics of the data. However, hand designing features requires expert knowledge and often assumptions are made about the nature of the dataset. Such assumptions negatively affect the performance of the designed features when testing on data collected from different sources.

Recently, feature learning methods have emerged as an alternative to hand-crafting and have offered better generalization for many computer vision tasks. In this chapter we will focus on deep learning which is a kind of feature learning approach, to perform object class segmentation. In particular, we investigate strategies for efficient pixelwise object class labeling of indoor scenes that combine both pretrained semantic features transferred from a large color image dataset and geometric features, computed relative to the room structures, including a novel distance-from-wall feature, which encodes the proximity of scene points to a detected major wall of the room. Several deep learning models are tested, which are designed to exploit different characteristics of the data. This includes feature learning at a coarse to fine level. Our results indicate that combining semantic and geometric features yields significantly improved results for object class segmentation. All the work in this chapter is a result of collaborating with Prof. Sven Behnke and Hannes Schulz from the University of Bonn.

7.1 Introduction

Understanding complex scenes has gained much in importance as the applications of service robots for homes and offices is increasing. Dense structural description of the indoor scenes is vital for performing accurate analyses. To serve this purpose, the usage of RGB-D cameras are becoming ubiquitous, as they provide color images and dense depth maps of the scene. Tasks such as "picking up objects" and "planning manipulation actions" (Dragan et al., 2011; Martínez et al., 2015; Husain et al., 2014a) are simplified once the precise location of objects in the scene is identified. One way to facilitate object localization is to perform pixelwise semantic labeling of the scene (Zhang et al., 2015). This involves identification and labeling of different object classes based on the

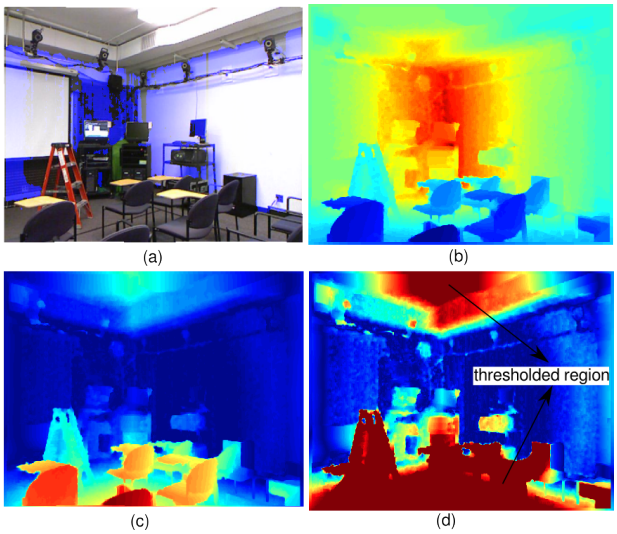

Figure 7.1: Illustrating the distance-from-wall feature. (a) Color image with its blue channel replaced with the binary mask based on the bounding hull heuristic from (Silberman and Fergus, 2011). The walls are detected from the bounding hull region. (b) Depth image, (c) and (d) show the minimum distance of each point in the scene from the walls before and after thresholding, respectively.

semantics of the scene. Recently, Convolutional Neural Networks (CNNs) have shown impressive results for semantic labeling (Eigen and Fergus, 2015; Long et al., 2015). The architecture of the CNN together with the used input features are important factors determining the quality of learned scene semantics. This chapter addresses these aspects with proposed novelties to improve the accuracy in semantic labeling.

The accuracy of semantic labeling depends on the consistency of the model description of the scenes under naturally occurring variations such as camera pose, lighting conditions and position of the objects. Object surfaces such as "floors" and "walls" are easy to identify and segment, as they usually follow a similar pattern. Movable objects such as "small structures", "furniture" and "wall hangings" are harder to identify. This is also evident by comparing the individual class labeling accuracies of different approaches (see Table 7.4). In order to mitigate this, Silberman and Fergus (2011) proposed a depth normalization scheme where the furthest point is assigned a relative depth of one. Using the depth normalization scheme improved the segmentation results, which motivated us to explore further the explicit modeling of the indoor environments.

For indoor environments, the height of an object above the ground plane gives a strong indication for the corresponding object class. For example, a "tv" is usually placed at a higher position from the ground plane than a "bed" or a "sofa". Schulz et al. (2015) showed that this feature improves the results for object class segmentation problem. We use the HHA representation of Gupta et al. (2014), which encodes the depth data into three channels, height above the ground, horizontal disparity, and angle of the normals with the inferred gravity direction. This encoding has been demonstrated to give good results for object detection and labeling tasks.

As the indoor scenes are always captured in a confined environment, i.e., in the presence of surrounding walls, we propose to exploit this structural information. Based

on a bounding hull heuristic developed by Silberman and Fergus (2011), we construct a novel feature that we name *distance-from-wall*, which indicates the proximity of scene points to some major detected room wall. This feature is defined as the minimum point-to-plane distance between a scene point and the planes detected at the outermost region of a scene, saturated beyond a certain distance threshold. Figure 7.1 shows an example of the computed distances from two walls for a sample taken from the NYU v2 dataset (Silberman et al., 2012). It can be observed that—compared to the original depth image in Fig. 7.1(b)—, the objects closer to the wall become more distinguishable after thresholding in Fig. 7.1(d). Wall distances greater than a fixed threshold are clipped, since we can assume that their precise wall distance is not informative. The proposed distance-from-wall feature facilitates the detection of objects such as "windows", "wall hangings" and "tv" which are usually found in close proximity to the walls. As a result, we observed an improvement in the overall object class segmentation accuracies.

One issue arising when training CNNs on RGB-D inputs is the limited size of the available RGB-D data sets. It has been shown that semantic CNN features obtained by training classification tasks on large data sets can be transferred to related tasks, such as object detection (Girshick et al., 2014; Sharif Razavian et al., 2014), subcategorization, domain adaptation, scene recognition (Donahue et al., 2014), attribute detection, and instance retrieval (Sharif Razavian et al., 2014). This transfer of pretrained semantic features proved also to be useful for RGB-D object categorization, instance recognition and pose estimation (Schwarz et al., 2015), and for the task of RGB-D object-class segmentation (Eigen and Fergus, 2015).

Our objective is to build a robust CNN architecture that predicts a semantic label for each pixel in the scene. We train CNN models with two different sets of pooling sizes end-to-end with the color image, the HHA encoding (Gupta et al., 2014), and our proposed distance-from-wall feature. We transfer pre-trained semantic features from a CNN model trained on the ImageNet dataset [1]. The parameters of the networks are learned so that they minimize the pixelwise cross-entropy loss between the predicted labels and the ground truth. We demonstrate the effectiveness of our approach by evaluating each of the proposed modalities separately and also comparing it with the other state-of-the-art approaches on the widely used NYU v2 dataset (Silberman et al., 2012).

The highlights of this chapter are:

- proposal of a new feature termed *distance-from-wall*,
- a novel CNN architecture using two different pooling sizes, yielding improved results, and
- evaluation and comparison with other state-of-the-art approaches, showing improved overall performance.

7.2 Related Work

The conventional approach to semantic labeling is carried out in multiple stages (Zhang et al., 2015; Cadena and Kosecka, 2014; Xiong et al., 2011; Hermans et al., 2014; Wolf et al., 2015; Müller and Behnke, 2014; Husain et al., 2014b). This involves presegmenting the scene into smaller patches followed by feature extraction and classification. The final

[1] http://www.image-net.org/

classification results are dependent on the results obtained at each stage of the approach. Another way is to train a deep CNN in an end-to-end fashion, i.e., directly from input pixels to semantic labels (Eigen and Fergus, 2015; Schulz et al., 2015; Farabet et al., 2012).

Zhang et al. (2015) performed a multiscale segmentation of image and point cloud followed by extraction of feature vectors. The feature vectors were used to do a unimodal classification and the results were further refined using a pairwise CRF (Conditional Random Field) to enforce spatial consistency. Handcrafted features such as "area," "diameter" and "orientation" were used to identify different features. However, feature learning from combined raw data and hand-crafted features often yields better results as it exploits both the hidden cues and human knowledge.

Wu et al. (2014) and Hermans et al. (2014) presegment a scene and afterwards build a model that exploits their semantic relations. Wu et al. (2014) used a CRF-based model to relate pixel-wise and pair-wise observations to labels for hierarchical semantic labeling. Hermans et al. (2014) used a randomized decision forest for semantic segmentation, where the results were further refined using a dense CRF. Similarly, segmentation followed by a random forest classification to initialize the unary potentials of a CRF was proposed by Wolf et al. (2015).

Schulz et al. (2015) trained a deep CNN using image patches as input to the network, where the patch size was adjusted according to the measured depth of the patch center. This increased scale invariance and led to improved object class segmentation results.

In order to increase scale invariance in deep CNNs, Farabet et al. (2012) and Eigen and Fergus (2015) used input at multiple scales. A simplified version of the histogram of oriented gradient (HOG) descriptor applied to the depth channel provided depth information to the CNN of Höft et al. (2014).

To increase the spatial accuracy of semantic segmentation, Eigen and Fergus (2015); Long et al. (2015) proposed two different CNN models. Eigen and Fergus (2015) divided a CNN into three sub-networks which gradually predicted output from a coarse to fine level. The network is initialized with ImageNet-trained AlexNet (Krizhevsky et al., 2012). Additionally, loosely related computer vision tasks—estimating depths and surface normals—were optimized by adjusting the loss function. Long et al. (2015) combined upsampled predictions from intermediate layers with the final layer which lead to more refined results. A single-image classification network was adapted to a fully convolutional network and fine tuned for semantic segmentation.

7.3 Problem Formulation

Given a color image and a dense depth map X of a scene, our goal is to obtain a label $\hat{y}_p \in \mathcal{C}$ for each pixel location $\mathrm{x}_p \in X$ that corresponds to the object class at the pixel location.

In our task, we deal with natural indoor scenes which are usually unbalanced with respect to the size and number of objects. For example, the number of pixels belonging to the floor class is much greater than that of those in the furniture class. Hence we use a weighted, multiclass cross entropy loss function (Eigen and Fergus, 2015),

$$L = -\sum_{i \in X} \sum_{b \in \mathcal{C}} \alpha_b c_{i,b} \ln(\hat{c}_{i,b}),$$

where

$$\alpha_b = \text{median-freq}(\mathcal{C}) / \text{freq}(b),$$

$\hat{c}_{i,\cdot}$ is the predicted class distribution at location i, and $c_{i,\cdot}$ is the respective ground truth class distribution. The factor α weighs each class according to its frequency with which it appears in the training set, and median-freq($\mathcal{C}$) is the median of all class frequencies.

7.4 Approach

We use a convolutional neural network in two stages, as illustrated in Figs. 7.2 and 7.3. In the first stage, we train the network with two different sets of pooling-operator sizes. In the second stage, we concatenate the feature maps of the last layer of the two networks from the previous stage and train the network again. This two-stage approach yields refined results (see Sec. 7.4.1). Finally, we enrich the set of concatenated features with geometric wall proximity information from the distance-from-wall feature (Sec. 7.4.2).

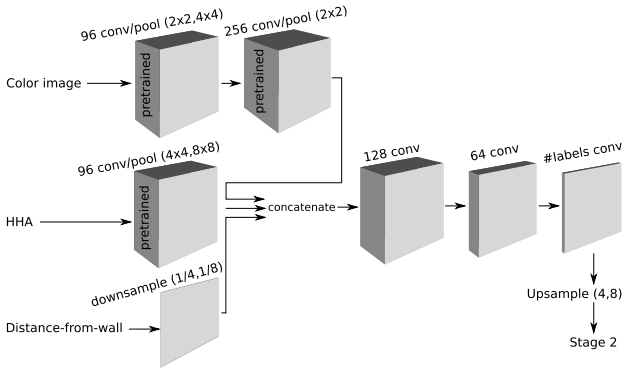

Figure 7.2: First stage of our proposed model architecture for object class segmentation. Inputs are the color image, the HHA encoding (Gupta et al., 2014) and the distance-from-wall feature (Sec. 7.4.2). Layer 1 (96 feature maps) and layer 2 (256 feature maps) for color image and layer 1 for HHA (96 feature maps) have filters pretrained on the ImageNet dataset which are not changed during training. Afterwards, the feature maps together with the distance-from-wall feature are concatenated and fed to a 3-layer trainable CNN. The output layer has the same number of feature maps as the number of object class labels. The output maps for the two different pooling sizes are used as input for Stage 2.

7.4.1 Network architecture

Our network takes three inputs, i.e., color image, HHA encoding and distance-from-wall feature. The color image is passed through a stack of five convolutional layers. The HHA encoding and distance-from-wall feature are processed by only four and three convolutional layers, respectively. This scheme is illustrated in Fig. 7.2. The weights of the first two layers for color image and the first layer for HHA encoding are transferred from the OverFeat network (Sermanet et al., 2014). This network was designed by

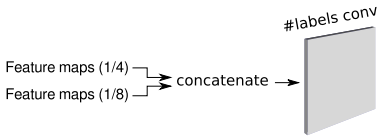

Figure 7.3: Second stage of our proposed CNN architecture for object class segmentation. The output feature maps obtained after training the CNN in the first stage (Fig. 7.2) are used as input. The pooling operation in Stage 1 is done with two sets of sizes. Hence, two sets of output feature maps are obtained which are here referred to as (1/4) and (1/8). These maps are concatenated and used as an input for training a single layer network. The output layer for this network has the same number of feature maps as the number of object class labels.

the CILVR Lab at NYU[2] and was trained on the ImageNet dataset for color image categorization. The weights of these two layers are kept fixed during our training and serve as semantic feature extractor. The network also contains pooling layers, a dropout layer and is divided into four configurations as described in Table 7.1. We train the network with two different pool sizes after the first convolutional layer. Hence, the final feature map size in configuration A is either four times or eight times smaller, depending on the pool size. The same applies to configuration B. The configurations A, B do not need training as the filters are transferred from the OverFeat network. The configuration C contains only the distance-from-wall feature. The outputs from the configurations A, B and C are concatenated and fed to configuration D, which we train using the labeled training examples.

The networks are trained for two different sets of pool sizes. The final output from configuration D is concatenated and trained again as shown in Fig. 7.3 before the upsampling. A single convolutional layer with the number of filters equal to the number of object classes and a size of (3×3) is learned. By concatenating networks with different pool sizes, we exploit invariance to local deformations while preserving information on spatial location, which increases the segmentation accuracy.

7.4.2 Distance-from-wall

We have devised a simple yet robust heuristic to detect the walls at the outermost region of an indoor scene. This involves segmenting the pointcloud and picking up those segments that lie within the outermost boundary region. Afterwards, the two largest segments are selected and a planar surface model is used to compute the distance from each point to those planar surfaces. The steps are illustrated in Fig. 7.4 and are detailed below.

Segment the point cloud

We partition the point cloud into surface segments using two different approaches. The first approach fits planes to the point cloud using RANSAC (see Fig. 7.4(c)). The second approach segments the 3D points based on quadratic surface fitting as described by Husain et al. (2015) (see Fig. 7.4(d)). We use these two different approaches, because depending on the complexity of the scene, one approach performed better than the other.

[2] http://cilvr.nyu.edu/doku.php

Table 7.1: Convolutional neural network configurations

Layer	Filter Size	Stride	No. of maps	Map size
Configuration A				
Input (color)	-	-	3	image_size
Conv1	11×11	1	96	image_size
Pool1	2×2, 4×4	2,4	96	image_size/(2,4)
Conv2	11×11		256	image_size/(2,4)
Pool2	2×2	2	256	image_size/(4,8)
Configuration B				
Input (HHA)	-	-	3	image_size
Conv1	11×11	1	96	image_size
Pool1	4×4, 8×8	4,8	96	image_size/(4,8)
Configuration C				
Input (Dist-from-wall)	-	-	1	image_size/(4,8)
Configuration D				
Input (Conf. A+B+C)	-	-	256+96+1	image_size/(4,8)
Conv1	11×11	1	128	image_size/(4,8)
Dropout (0.25)	-	-	-	-
Conv2	11×11	1	64	image_size/(4,8)
Conv3	11×11	1	No. of classes	image_size/(4,8)
Upsample	-	-	No. of classes	image_size

The method by Husain et al. (2015) performed better in complex scenes containing mostly non-planar surfaces, whereas RANSAC performed better when the scene was dominated by planar surfaces.

Detect the outermost boundary

In order to detect the outermost boundary, we use the technique described by Silberman and Fergus (2011), i.e., if the depth of a point is within 4% of the maximum depth within each column of the image grid then it is marked as the boundary region. Fig. 7.4(a) shows the boundary region shaded in blue color. We only take the segments that lie in the outermost boundary as possible candidates for walls as shown in Figs. 7.4(e) and (f).

Select the largest two segments when viewed from the top

To generate an approximate top view, we assume that the vertical camera-axis always points towards the upwards direction. We select at most two segments from the segmented point cloud that have the largest triangular width and are not coplanar, when viewed from the top. Afterwards we select the segments that have the largest sum

of widths. Hence, among Figs. 7.4(g) and (h), the two segments from Fig. 7.4(g) are selected.

Computing the distance from the wall feature

Once the two segments are selected, we compute for all scene points the minimum point-to-plane distance using a planar-surface equation. Since in each scene we have at most two planes yielding a point-plane distance value each (see Figs. 7.4(i) and (j)), we take the minimum to get the final distance-from-wall (see Fig. 7.4(k)),

$$D_{x \in \text{pixel}} = \min(\pi_x^1, \pi_x^2),$$

where π_x^1 and π_x^2 are the point to plane distances of point x to plane π^1 and π^2 respectively. We use a threshold value of one meter. All points whose value exceeds the threshold are clipped as shown in Fig. 7.4(l). The rationale behind this particular value is that the objects within a 1 meter proximity of walls are likely to have depth readings differing slightly, but distinctly from the wall itself, such as "lamps", "wall hangings" and "cupboard". Such objects are highlighted from the rest of the scene after applying this threshold. Objects further away than one meter do not typically occur in specific distances to the wall and the feature becomes meaningless.

7.5 Experiments

In this section we describe the evaluation results on the NYU v2 dataset. We use the four object classes (Silberman et al., 2012) in Sec. 7.5.1 and the 13 object classes (Couprie et al., 2013) in Sec. 7.5.2.

The dataset contains a total of 1449 samples of different indoor scenes. We use the training/test split as provided by the dataset authors. The parameters for gradient descent, i.e., the learning rate, momentum and the number of iterations are adjusted by first separating 10 % of the training examples and using them for validation.

In order to evaluate our approach, we use two common measures of performance, average pixel accuracy $\sum_i n_{ii} / \sum_i t_i$ and average class accuracy $(1/n_{cl}) \sum_i (n_{ii}/t_i)$, where n_{ii} is the number of correctly classified pixels for class i, t_i is the total number of pixels for class i, and n_{cl} is the number of classes.

In order to get an insight on the benefits of our proposed models, we evaluate different aspects separately. This includes the network from first stage, referred to as net-ABCD and without the distance-from-wall feature as net-ABD, the network from second stage referred to as net-ABCD-combined and also without the distance-from-wall feature as net-ABD-combined. We distinguish between the two upsampling factors for the different pooling sizes in the first stage as U4 and U8.

7.5.1 NYU v2 with 4 classes

We present the results for the NYU v2 dataset, using four classes as defined by Silberman and Fergus (Silberman et al., 2012). Table 7.2 shows a comparison of the individual labeling accuracies for each of the four classes. We get better results after combining the two networks, in the struct class (81.9% vs. 81.6% and 79.9%) and furniture class (72.8% vs. 66.6% and 72.0%). Table 7.3 shows a comparison of the average class accuracies and the average pixel accuracies, corresponding to the results in Table 7.2.

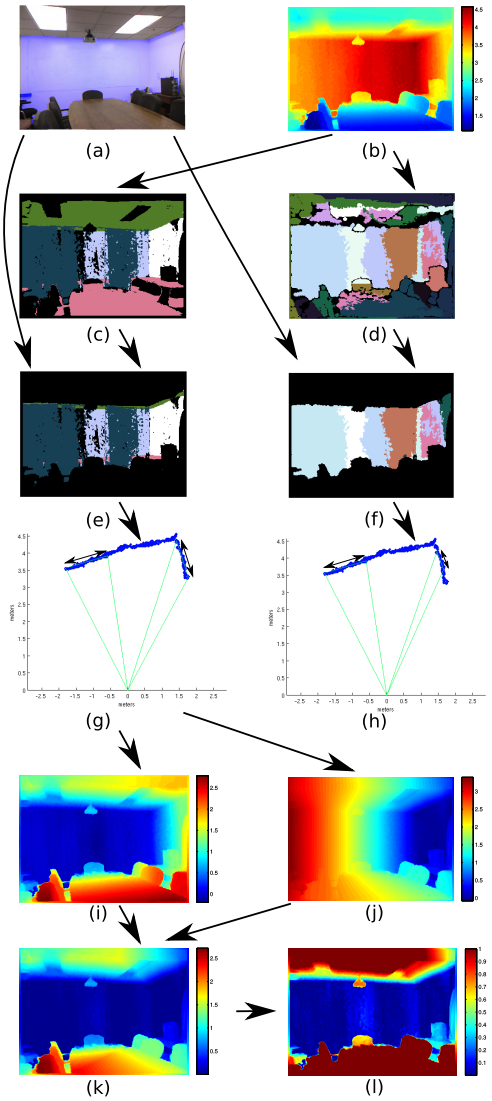

Figure 7.4: Illustrating the procedure for the computation of the distance-from-wall feature. (a) Color image with the boundary region shaded in blue, (b) depth image, (c) segmentation using RANSAC, (d) segmentation of depth image using the method proposed by Husain et al. (2015), (e) and (f) selecting only the segments from the boundary region, (g) and (h) are top views of the point clouds based on the two largest single-colored plane segments from (e) and (f), respectively, (g) is selected because it has the largest sum of the widths (the double sided arrows in (g) and (h)) of the enclosed segments, (i) and (j) are the distances of each point from the two planes detected based on the two segments in (g), (k) is the minimum distance for each point from both planes and (l) is the final distance feature after thresholding. The color image is shown for illustration only and not used in computing the distance feature.

The net-ABCD-combined shows improved overall results as compared to net-ABCD-U4 and net-ABCD-U8. Our results are competitive to the cascaded multi-scale CNNs

Table 7.2: Individual classes of NYU v2 (four classes).

Method	Accuracy (%)			
	floor	struct	furniture	prop
Couprie et al. (2013)	87.3	86.1	45.3	35.5
Khan et al. (2014)	87.1	**88.2**	54.7	32.6
Stückler et al. (2015)	90.7	81.4	68.1	19.8
Müller and Behnke (2014)	94.9	78.9	71.1	42.7
Wolf et al. (2015)	**96.8**	77.0	70.8	45.7
net-ABD-U4 (without dist-from-wall)	96.6	82.5	62.8	63.2
net-ABD-U8 (without dist-from-wall)	94.9	78.1	75.9	60.5
net-ABCD-U4	95.9	79.9	72.0	63.5
net-ABD-combined (w/o dist-from-wall)	94.8	78.2	69.2	69.9
net-ABCD-U8	94.8	81.6	66.6	**70.3**
Eigen and Fergus (2015) (AlexNet)	93.9	87.9	**79.7**	55.1
net-ABCD-combined	95.0	81.9	72.8	67.2

Table 7.3: Overall performance on NYU v2 (four classes)

Method	Accuracy (%)	
	class avg.	pixel avg.
Couprie et al. (2013)	64.5	63.5
Khan et al. (2014)	69.2	65.6
Stückler et al. (2015)	70.9	67.0
Müller and Behnke (2014)	72.3	71.9
Wolf et al. (2015)	72.6	74.1
net-ABD-U4 (without distance-from-wall)	76.3	74.6
net-ABD-U8 (without distance-from-wall)	77.4	76.4
net-ABCD-U4	77.8	76.5
net-ABD-combined (without distance-from-wall)	78.2	76.5
net-ABCD-U8	78.3	76.4
Eigen and Fergus (2015) (AlexNet)	79.1	**80.6**
net-ABCD-combined	**79.2**	78.0

approach (Eigen and Fergus, 2015). However, because of cascading different networks, the latter model required training in two steps. Additionally, the feature learning at multiple scales approach in (Eigen and Fergus, 2015) seems to be beneficial, when compared to our single scale model.

Figure 7.5 shows selected examples from the test set of the NYU v2 dataset. Results obtained with and without the distance-from-wall feature are shown. It can be observed that the furniture (brown color) and the prop (pink) class are better segmented. For example, in Fig. 7.5(2c) it can be seen that the "tv" is easily distinguishable in the distance-from-wall feature and it is better segmented (see Fig. 7.5(2d) and (2e)) when this feature is used.

7.5.2 NYU v2 with 13 classes

We present the results for the NYU v2 dataset, using 13 classes as defined by Couprie et al. (Couprie et al., 2013). Table 7.4 shows a comparison of the individual labeling accuracies for each of the 13 classes. It can be observed that our approach performs better

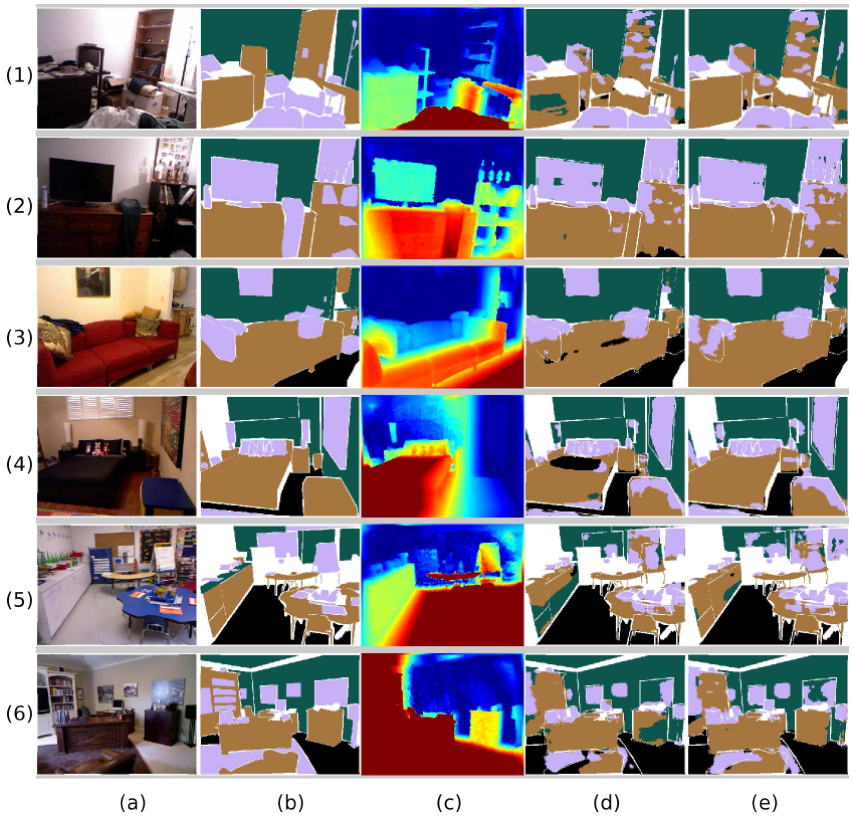

Figure 7.5: Selected test set examples showing the improved labeling after using our proposed feature, (a) color image, (b) ground truth labeling, (c) distance-from-wall, (d) predicted labels without distance-from-wall and (e) predicted labels with distance-from-wall. White color in Figs. (b), (d) and (e) represents the unknown label.

Table 7.4: Individual class labeling accuracy for NYU v2 (13 classes)

	Accuracy (%)												
Method	bed	books	ceiling	chair	floor	furniture	objects	picture	sofa	table	tv	wall	window
Couprie et al. (2013)	30.3	31.7	33.2	44.4	68.0	28.5	10.9	38.5	25.8	18.0	18.8	**89.4**	37.8
Hermans et al. (2014)	**68.4**	45.4	83.4	41.9	91.5	37.1	8.6	35.8	28.5	27.7	38.4	71.8	46.1
net-ABD-U4 (without distance-from-wall)	21.1	22.3	94.4	19.5	93.2	69.9	49.2	68.9	27.3	41.6	39.5	62.3	53.4
net-ABD-U8 (without distance-from-wall)	48.0	32.3	93.3	37.1	94.7	77.4	48.0	45.1	44.3	41.7	30.6	58.3	71.2
net-ABCD-U4	52.5	42.5	**93.3**	32.1	93.3	**67.2**	48.4	63.5	**53.2**	34.1	25.8	66.2	58.0
Wolf et al. (2015)	58.2	45.3	92.8	57.7	**97.5**	57.3	37.4	32.3	49.8	**51.8**	26.4	74.4	43.2
net-ABD-combined (without distance-from-wall)	42.2	37.6	92.9	34.3	94.7	63.9	50.8	**70.0**	41.5	53.6	42.5	70.4	61.0
Eigen and Fergus (2015) (AlexNet)	57.7	39.9	77.6	**71.1**	95.9	64.1	**54.9**	49.4	45.8	45.0	25.2	87.9	57.6
net-ABCD-U8	51.9	**46.5**	91.8	41.0	94.8	66.7	44.9	61.7	49.4	49.0	41.6	73.5	60.5
net-ABCD-combined	44.1	42.9	92.7	38.5	95.2	66.6	53.9	63.5	46.8	49.4	**42.6**	74.7	**63.5**

for the smaller object categories that are usually closer to the wall such as “picture”, “tv” and “window”. Table 7.5 shows a comparison of the average class accuracies and the average pixel accuracies corresponding to the results in Table 7.4. Similar behavior can be observed, as was the case of four classes, i.e., better results with net-ABCD-combined as compared to net-ABCD-U4 and net-ABCD-U8.

Table 7.5: Overall performance for the NYU v2 (13 classes)

Method	Accuracy (%)	
	class avg.	pixel avg.
Couprie et al. (2013)	36.2	52.4
Hermans et al. (2014)	48.0	54.2
net-ABD-U4 (without distance-from-wall)	50.9	58.3
net-ABD-U8 (without distance-from-wall)	54.9	61.1
net-ABCD-U4	56.2	62.5
Wolf et al. (2015)	56.9	64.9
net-ABD-combined (without distance-from-wall)	58.1	64.1
Eigen and Fergus (2015) (AlexNet)	59.4	**70.5**
net-ABCD-U8	59.5	65.7
net-ABCD-combined	**59.6**	66.4

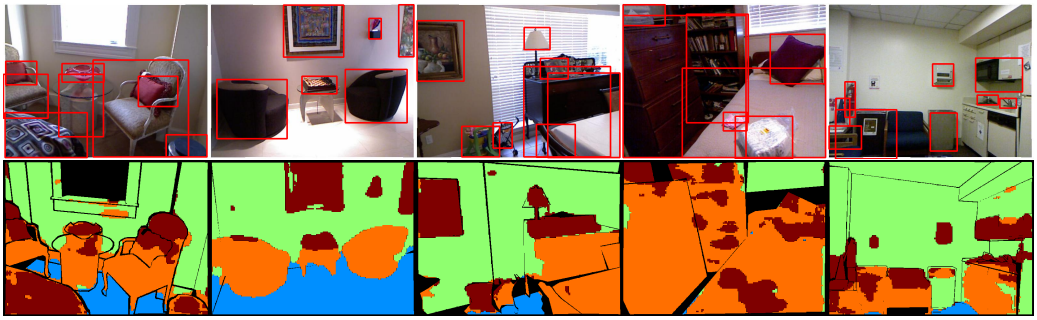

Figure 7.6: Top row: example of the successful candidates obtained after incorporating semantic segmentation into the method of Martín García et al. (2015) for four frames of the NYU Depth V2 Dataset. Bottom row: the corresponding semantic segmentation obtained in those frames.

7.6 Summary and Future Work

We have proposed a new model based on deep learning for the task of semantic object class segmentation. The model employs a multi-pooling architecture and takes the color image, the HHA encoding and a novel distance-from-wall feature as input. The distance-from-wall feature is able to successfully highlight objects that are in close vicinity to the walls. This enables the deep learning model to learn from a more detailed representation of a scene. Our extensive evaluation on the NYU v2 dataset demonstrates the effectiveness of our proposed feature by showing better overall object class segmentation results. In this chapter, we showed that features of relative position are helpful for semantic segmentation. In the future, we plan to increase invariance to camera position even further by taking into account additional properties such as 3D room layout.

Our approach can serve as an important prior for object discovery because it clearly separates the object classes. Figure 7.6 shows an example from (Martín García et al., 2016), in which a saliency-based generic objects candidates of Martín García et al. (2015) is improved by incorporating semantic segmentation into the pipeline.

Chapter 8

Action Recognition

We close the thesis with tackling the action recognition problem along the lines of robot perception. Particularly, in this chapter we will address the higher level computer vision task of recognizing actions in videos. The recognition of actions becomes extremely challenging in the real world. Often, there are limited constraints on otherwise visually important cues extracted from the background, camera motion and object appearances. The problem can be efficiently addressed by learning features from large-scale databases making it inherently aware of such variations. Fortunately, with the open source implementations, we can now leverage the power of deep convolutional architectures, trained on the largest annotated dataset of over a million images, i.e. ImageNet. We will show how to efficiently adapt a model learned from this dataset for recognizing video content.

8.1 Introduction

Building personal robots for tasks involving assistance and interaction with humans carries several challenges. One key challenge is to perceive and interpret dynamic human environments. This is necessary for the active engagement of the robot. Several attempts have been made to address the different perception aspects of such dynamic environments where the robot is meant to assist, such as tracking a hand-held object for grasping (Husain et al., 2014a), capturing human motion (Chan and Nejat, 2011), activity recognition (Chrungoo et al., 2014) and sensing the human behaviors (Kanda et al., 2010).

One important objective is the detection and recognition of daily human activities. Actions such as brushing hair, eating, drinking, chewing, sitting, walking, standing, etc., implicitly encompass the structure of a particular human environment. Successful recognition of these actions simplifies several tasks that are aimed for such robotic assistants. For example, assisting the elderly in timely caregiving (Liu et al., 2015; Schroeter et al., 2013), in the situation of accidents (Nakagawa et al., 2015) or in the daily life activities (Park et al., 2008).

Recognizing human activities for robots is conventionally tackled using a pipeline approach, by first (i) modeling the dynamics of changing environments using a graphical

model (Anand et al., 2013; Hu et al., 2014; Sung et al., 2012) or identifying descriptive features (Klaeser et al., 2008; Stefan Mathe, 2012; Laptev, 2005; Wang et al., 2014; Husain et al., 2014c), and then (ii) performing classification (Schuldt et al., 2004; Scovanner et al., 2007). The first part requires extraction of motion information through some mechanism. Possible approaches include the computation of optic flow or motion modeling. However, recent benchmarks have revealed that there is no universally accepted model that could outperform others for all datasets (Wang et al., 2009b). The reason is that the statistics of datasets can be considerably different, and a particular model might perform better for one dataset than for another. Many spatio-temporal descriptors are extensions from single image descriptors such as SIFT3D (Scovanner et al., 2007), HOG3D (Klaeser et al., 2008) and SURF3D (Willems et al., 2008). However, such extensions also inherit the limitations in performance generalization as shown in (Le et al., 2011b), making clear the advantage of learned features over hand-crafted ones.

Deep Convolutional Neural Networks (CNNs) (Lecun et al., 1998) have emerged as a state-of-the-art solution for a wide range of computer vision problems, such as image segmentation/labeling (Girshick et al., 2014; Pinheiro and Collobert, 2014), object detection/localization (Oquab et al., 2014) and pose recovery (Ouyang et al., 2014; Jain et al., 2014). The main advantage over the conventional pipeline approaches is that CNNs can be trained end-to-end (from raw pixels to labels) in a fully supervised way. One drawback of fully supervised deep learning is that it requires a huge number of labeled training examples (Jaderberg et al., 2014).

Recently, it has been shown that a CNN model trained from a large dataset can be transferred to other visual recognition tasks with limited training data and thereby leading to higher accuracy and shorter training period (Oquab et al., 2014; Donahue et al., 2014). Since single-image input-based models that have been trained over a million labeled images are now readily available (Jia et al., 2014), we see attempts to exploit these networks in the video domain (Simonyan and Zisserman, 2014a; Karpathy et al., 2014). However we observe limited success when learning directly in the temporal domain.

We also notice that weakly annotated video data is becoming prevalent as time goes by. For example, 300 hours of video are uploaded to Youtube every minute[1]. Such abundance of video data opens up the opportunity to exploit the infinite space of possible actions in the context of human action recognition (Karpathy et al., 2014). Visual recognition methods have to interpret video data displaying a large degree of variability and complexity in order to arrive at a semantic description, i.e., the action class of the recorded scene.

We propose to recognize human actions using the transfer learning technique. A pretrained single-image recognition model is adapted for videos by temporally concatenating the output of its deepest spatial convolution layer. The input to the model are the individual frames of the video. Afterwards, the concatenated output is used as an input to a network comprising 3D convolutions that we train.

The feature representation becomes more abstract as we go deeper in a network thereby obscuring the locally occurring temporal changes in a video. This poses a limitation to the temporal features that the network learns from the concatenated output. We overcome this limitation by combining the output of our learned network with an-

[1] https://www.youtube.com/yt/press/statistics.html

other pretrained model (Tran et al., 2015) which employs 3D convolutions from the beginning. The complementary nature of the two features becomes evident from the improved recognition accuracy in our experiments.

The combined output contains fewer trainable parameters thereby allowing us to use a more efficient optimization method (L-BFGS) (Le et al., 2011a). Our model does not require any pre-computation of features such as optic flow or any other domain-specific processing, thereby making it generic and computationally efficient.

The highlights of this chapter are:

- the introduction of a concatenation scheme in the temporal domain to extend the usage of pretrained models learned from a single image to the video domain,
- combining our learned network with another action recognition model, which yields improved results as compared to the individual networks, and
- evaluation and comparison with commonly used benchmarking video datasets.

8.2 Related Work

Several action recognition methods have been proposed in the past. We roughly group them into two categories. First is the conventional pipeline approach (descriptor followed by a classifier) (Wang and Schmid, 2013; Sadanand and Corso, 2012; Scovanner et al., 2007; Klaeser et al., 2008; Willems et al., 2008; Aksoy et al., 2011) and second is the convolutional model (Le et al., 2011b; Baccouche et al., 2012; Taylor et al., 2010; Ji et al., 2013; Karpathy et al., 2014) which is the basis of our approach.

In (Wang and Schmid, 2013), improved dense trajectories are produced by reducing the camera motion effect, which is estimated using the SURF descriptor (Bay et al., 2008). However, for recognizing human actions, inconsistent matches from SURF are removed by exploiting domain knowledge, i.e., by adding a human detector. A higher-level representation of activities, named as "action bank," combined with a linear SVM classifier is proposed in (Sadanand and Corso, 2012).

Another way of representing actions is through spatio-temporal segmentation of dynamic scenes. The segmented surfaces and how they change over time gives cues about the kind of manipulation actions which are being carried out. The manipulation actions can be encoded in the form of "Semantic Event Chains" (Aksoy et al., 2011). These event chains represent the spatial relations between objects in the scene. Any change in the spatial relation serves as a decisive key point through which a manipulation could be defined. Similarly, temporal segmentation of a video into multiple events is proposed in (Zhang et al., 2014).

An unsupervised learning method based on convolution and stacking has also been proposed (Le et al., 2011b). The convolved output of arbitrarily sized videos is made constant by dimensionality reduction using principal component analysis. The time-efficient dimensionality reduction for long video clips has a relatively larger memory requirement (up to 32 GB). In (Baccouche et al., 2012), a spatio-temporal sparse auto-encoder is trained in an unsupervised way for classifying video sequences. The convolutional gated Restricted Boltzmann Machine architecture (Lee et al., 2009) has been extended to 3D to learn relations between pairs of adjacent images in videos and is used to extract features for activity recognition (Taylor et al., 2010).

A 3D CNN model has also been previously proposed (Ji et al., 2013). In this model, features are learned simultaneously in the spatial and temporal dimensions by perform-

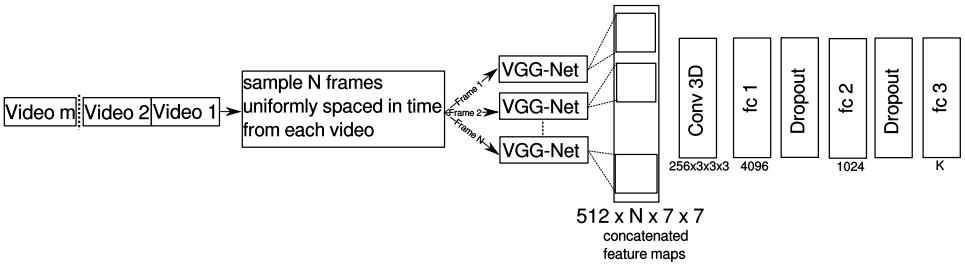

Figure 8.1: Illustration of the network. We use the output from layer 16 of the VGG-Net (Table 1 in (Simonyan and Zisserman, 2014b)), as a descriptor. The output is concatenated to form 512, 3D feature maps. The 3D feature maps are used as input for the network consisting of a volumetric convolutional layer followed by two fully-connected layers.

ing 3D convolutions. The model is applied to real-world environments to recognize human actions. However, other than the raw images, a set of hardwired kernels is created to generate the gradients and optic flow which should be learned by the proposed convolutional network. In addition, a human detector is introduced and foreground extraction is performed. On the contrary, we feed the raw image data directly to our network and do not compute any handcrafted feature.

In (Simonyan and Zisserman, 2014a), a two-stream network is proposed, where each frame of the video is used as an individual image during training. One stream is trained on raw images and the other is trained with optical flow fields computed from the consecutive video frames. Recognition is attained using a score aggregation strategy across all the video frames of both streams.

Using pretrained models is also proposed in (Karpathy et al., 2014; Zha et al., 2015; Tran et al., 2015). Our model could be categorized as the *late fusion model* from (Karpathy et al., 2014), in which multiple networks are fused in the final fully connected layers. However, we use a different network architecture which is pretrained on a single-image database instead of a video database, thereby reducing the computational cost. We get significantly better results without requiring fine tuning of the pretrained models. Along the same lines, different aggregation strategies for per frame image-based features is investigated in (Zha et al., 2015).

Our approach is closely related to Tran et al. (2015). A 3D convnet is defined with convolutional kernels up to the first 8 layers. However, we use 3D convolutional kernels after extracting the output from a very deep pretrained model and thereby learn features in the temporal domain at a higher level of abstraction.

8.3 Problem Formulation

Given a set of videos, obtain a label for each video characterizing its content. The video can be of arbitrary spatial and temporal dimensions.

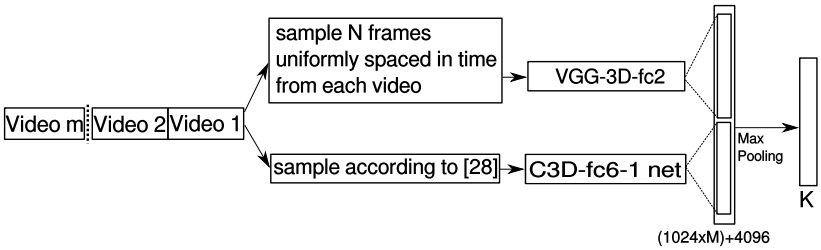

Figure 8.2: Illustration of how the different network outputs are combined, where VGG-3D-fc2 refers to the fc2 layer in Fig. 8.1

8.4 Approach

We use a single-image convolution model for individual frames of video data and perform volumetric convolution at a higher level of abstraction by temporally concatenating the output. The method is illustrated in Fig. 8.1. Here, K is the number of action categories. In this way we are able to initialize our network with the parameters learned from the ImageNet dataset (Krizhevsky et al., 2012). Additionally, we freeze the learned network parameters up to the second-last fully-connected layer, combine the output with another pretrained network and build a new softmax model. We refer to the CNN architecture (19-layer network in (Simonyan and Zisserman, 2014b)) as VGG-Net.

8.4.1 Feature map concatenation

We train a network which takes as input 3D feature maps. These feature maps are the outputs from layer-16 of the 19-layer network defined in (Simonyan and Zisserman, 2014b) and trained on the ImageNet dataset. Layer-16 is the last spatial convolution layer in (Simonyan and Zisserman, 2014b). This gives us a high-level feature descriptor in the spatial domain. Afterwards we add one 3D convolutional layer followed by three fully-connected layers. The K-way softmax function is applied to the output of the last fully-connected layer, where K is again the number of action categories. In total, our network contains 20 layers and we train only the last 4 layers. We use a dropout regularization ratio of 0.5 for the fully-connected layers.

We take $N = 30$ uniformly spaced frames from each video as input to the network. Each image is assigned the same label as its corresponding video. The network is trained using stochastic gradient descent and use the same momentum as in (Krizhevsky et al., 2012). The learning rate is adjusted to get the maximum accuracy in a minimum number of iterations on a held-out validation set from the training set.

8.4.2 Combining multiple networks

A deep network learns different features at each level of the layer hierarchy. The activations in the initial layers tend to be more sensitive to edge-like patterns and corners within their receptive field, whereas activations at deeper levels have larger receptive fields and capture more complex invariances (Zeiler and Fergus, 2014).

Our network learns changes that occur in the temporal domain at a more abstract level because we temporally concatenate the output of the convolutional layer-16 from the pretrained network of (Simonyan and Zisserman, 2014b). Hence, our model lacks

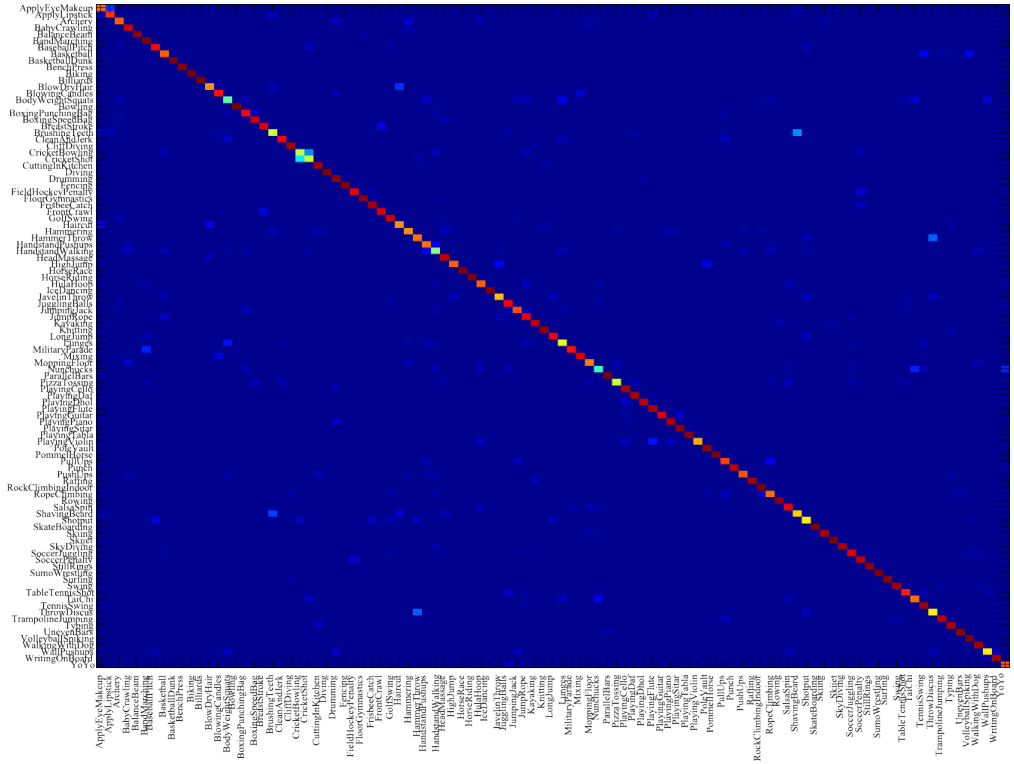

Figure 8.3: Confusion matrix for the UCF-101 dataset accumulated for all three splits

learning in the temporal domain from locally occurring changes. In order to palliate this deficiency, we concatenate the fully connected layer-9 feature vectors with a length of 4096, extracted from another deep network that was trained in 3D from the beginning (Tran et al., 2015). The model contains 8 3D-convolution, 5 max-pooling, and 2 fully-connected layers. Deeper 3D convolution layers are not possible, due to GPU memory restrictions. The model is trained on the Sports-1M dataset (Karpathy et al., 2014) which contains about 1 million videos of different sports action categories.

Figure 8.2 shows the combination scheme of the two feature maps. We concatenate the output of the fully-connected layers for the same video, hence having the same action category. We also augment the data by cropping $M = 10$ patches from each frame of the video for the VGG-3D network, hence the output feature dimension is $4096 + (1024 \times M)$. After concatenation, we perform max-pooling to reduce the feature dimension and afterwards build a new softmax model. Since we are learning the parameters for the softmax layer only, we can use a more efficient optimization approach instead of stochastic gradient descent. We use an off-the-shelf implementation[2] of L-BFGS which has been shown to yield better results when the number of trainable parameters is small (Le et al., 2011a).

[2] http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html

Table 8.1: Average accuracy on the UCF-101 dataset (3-fold).

Algorithm	Accuracy
CNN with transfer learning (Karpathy et al., 2014)	65.4%
LRCN (RGB) (Donahue et al., 2015)	71.1%
Spatial stream ConvNet (Simonyan and Zisserman, 2014a)	72.6%
LSTM composite model (Srivastava et al., 2015)	75.8%
Our approach (VGG-3D)	**79.1%**
C3D (1 net) (Tran et al., 2015)	82.3%
Temporal stream ConvNet (Simonyan and Zisserman, 2014a)	83.7%
C3D (3 nets) (Tran et al., 2015)	85.2%
Combined ordered and improved trajectories (Murthy and Goecke, 2013)	85.4%
Stacking classifiers and CRF smoothing (Karaman et al., 2013)	85.7%
Improved dense trajectories (Wang and Schmid, 2013)	85.9%
Improved dense trajectories with human detection(Wang et al., 2015)	86.0%
Our approach (VGG-3D + C3D-fc6-1 net)	**86.7%**
Spatial and temporal stream fusion (Simonyan and Zisserman, 2014a)	88.0%

8.5 Experiments

We evaluate our approach on two publicly available benchmarking datasets, UCF-101 (Soomro et al., 2012) and HMDB (Kuehne et al., 2011). These datasets are challenging because many video samples include camera motion as well as a dynamic background. We use the same evaluation protocol as proposed by the respective authors and provide an in-depth analysis of our approach using the UCF-101 dataset as a test case. Additional qualitative results for both the datasets are available at `http://www.iri.upc.edu/people/shusain/actionrecognition.html`

The network is able to take only fixed-size input frames, hence we resize all the videos so that the maximum dimension is 256 pixels and crop 10 patches of size 224×224 pixels according to the data augmentation scheme as proposed in (Krizhevsky et al., 2012). Furthermore, we separate 10% percent of the samples from the training data and use them as validation data. Such data are needed to determine the number of iterations needed for stochastic gradient descent.

8.5.1 UCF-101 dataset

The UCF-101 (Soomro et al., 2012) dataset contains 13,320 labeled video samples with 101 action categories. We use the 3-way train/test split as provided by the authors. Table 8.1 shows a comparison with other approaches. Compared with the baseline (Karpathy et al., 2014) we observe a considerable improvement. Not surprisingly, we see improved results as also compared to (Tran et al., 2015). This shows the complementary nature of the high-level (layer-19) and low-level (layer-6) features. It should be noted that we use the output from the layer-9 activation (C3D 1 net) and concatenate it with our trained model as described in Fig. 8.2, i.e., concatenating two networks, as opposed to (Tran et al., 2015), where the output from 3 networks that have been trained differently is combined. Our results are closer to (Simonyan and Zisserman, 2014a), where optical flow needs to be computed. However, the calculation of optical flow leads to a significant computational overhead. As shown in (Tran et al., 2015), Brox optical flow used in (Simonyan and Zisserman, 2014a) takes 0.85-0.95s per image pair which

Table 8.2: Convnet accuracy under different settings for UCF-101 dataset.

Scenario	Accuracy
Fine tune top 3 layers (Sports 1M - pretrained) (Karpathy et al., 2014)	65.4% (3 fold)
Fine tune all layers (Sports 1M - pretrained) (Karpathy et al., 2014)	62.2% (3 fold)
Spatial AlexNet-stream (pretrained and last layer) (Simonyan and Zisserman, 2014a)	72.7% (1 fold)
Spatial AlexNet-stream (pretrained and fine tuned) (Simonyan and Zisserman, 2014a)	72.8% (1 fold)
Spatial VGG-stream (pretrained and fine tuned)	71.4% (1 fold)
VGG-3D (pretrained and fine tuned)	75.5% (1 fold)
Spatial VGG-stream (pretrained and adaptation layers)	76.3% (1 fold)
VGG-3D (pretrained and adaptation layers)	80.0% (1 fold)
VGG-3D (pretrained and fine tuned) + C3D-fc6-1 net	83.5% (1 fold)
VGG-3D (pretrained and adaptation layers) + C3D-fc7-1 net	84.8% (1 fold)
VGG-3D (pretrained and adaptation layers) + C3D-fc6-1 net	86.7% (1 fold)

is 274x slower than C3D. Additionally, storing the raw flow fields for this dataset requires a disk space of 1.5 TB which needs data compression (Simonyan and Zisserman, 2014a). Figure 8.3 shows the confusion matrix accumulated for all the three splits. Comparing the confusion matrix with that resulting from the approach in (Simonyan and Zisserman, 2014a) (Fig. 5 in (Simonyan and Zisserman, 2014a)), it can be seen that the actions "CricketBowling" and "CricketShot" have similar levels of confusion, whereas our approach shows better results for the action "YoYo". Figure 8.6 shows the top-5 predictions for selected test sequences from the UCF-101 dataset (Soomro et al., 2012) with 101 action categories.

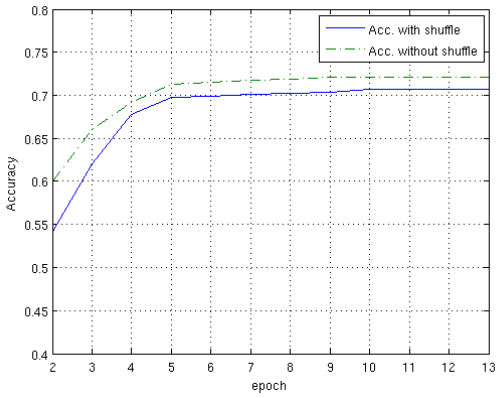

Figure 8.4: Comparing accuracy for shuffling video sample frames.

8.5.2 Evaluating different scenarios

We measure the performance of our spatial and spatio-temporal learning framework in different scenarios using the split-1 of UCF-101 dataset. Table 8.2 presents the evaluation under different settings along with the comparison to other approaches.

In our spatial VGG-stream we obtained the label for a video after averaging the scores for all the frames belonging to that video. All the layers were pretrained on the ImageNet dataset and fine tuned on the UCF-101 dataset, except the last layer which

Table 8.3: Average accuracy on the HMDB dataset (3-fold).

Algorithm	Accuracy
Spatio-temporal HMAX network (Jhuang et al., 2007)	22.8%
Spatial stream ConvNet (Simonyan and Zisserman, 2014a)	40.5%
Trajectory-Based Modeling (Jiang et al., 2012)	40.7%
Our approach (VGG-3D)	**46.9%**
Decomposing visual motion (Jain et al., 2013)	52.1%
Our approach (VGG-3D + C3D-fc6-1 net)	**53.9%**
Temporal stream ConvNet (Simonyan and Zisserman, 2014a)	54.6%
Improved dense trajectories (Wang and Schmid, 2013)	57.2%
Spatial and temporal stream fusion (Simonyan and Zisserman, 2014a)	59.4%

was initialized randomly because of different number of classes. We observed results similar to the spatial AlexNet-stream (Simonyan and Zisserman, 2014a). We found better results for spatial VGG-stream and VGG-3D when training only the adaptation, i.e., the newly added layers. Similar behavior was observed in (Karpathy et al., 2014), i.e., a drop in the accuracy when fine tuning all the layers. This is because training such a huge network with a small dataset results in overfitting. We observed the best result when training the adaptation layers only combined with the fc6 layer from C3D.

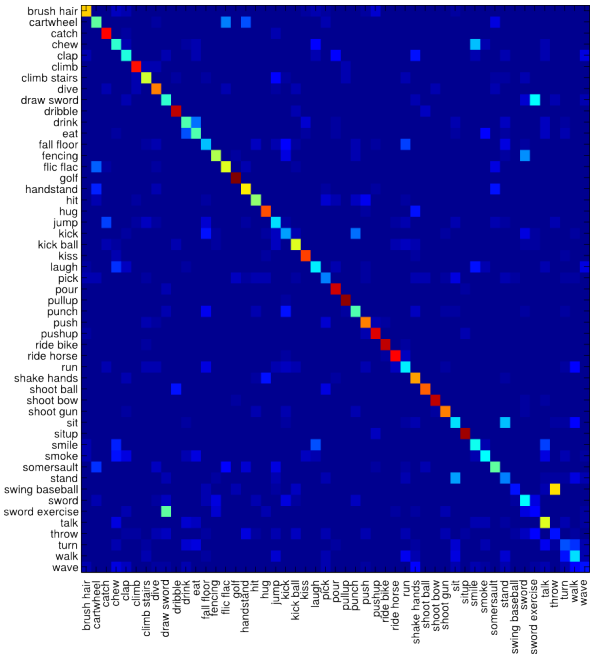

Figure 8.5: Confusion matrix for the HMDB dataset accumulated for all three splits

8.5.3 Learning from temporal information

Due to the concatenation of the feature maps in the temporal domain, the 3D kernels should also be able to exploit the temporal information in the video. Hence, if we randomly shuffle the video frames while training, we should see a drop in the accuracy

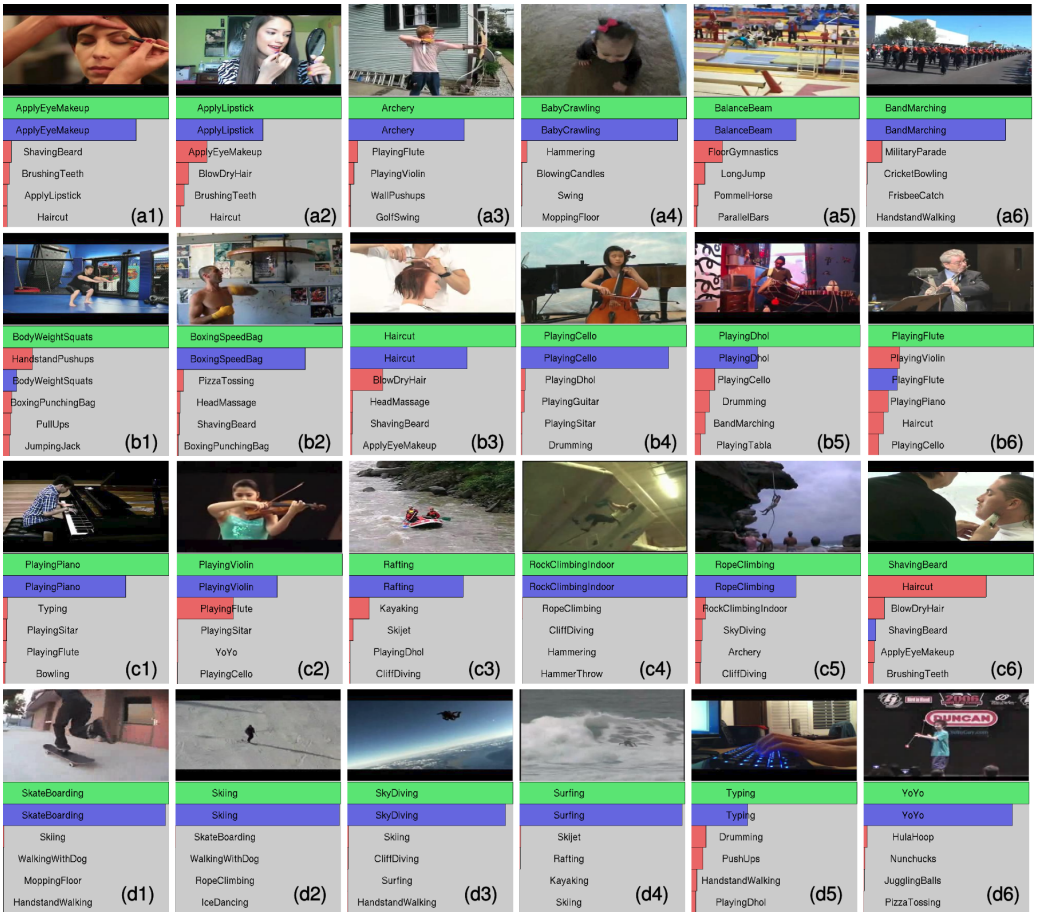

Figure 8.6: Top-5 predictions using our approach for selected test sequences from the UCF-101 dataset (Soomro et al., 2012) with 101 action categories. First row (green color) shows the ground-truth followed by predictions in decreasing level of confidence. Blue and red show correct and incorrect predictions, respectively.

due to temporal inconsistency. Figure 8.4 shows the drop in the accuracy averaged for two training sessions of the method described in Sec. 8.4.1 for split-1 of UCF-101 dataset.

8.5.4 HMDB dataset

The HMDB dataset (Kuehne et al., 2011) contains 6,849 labeled video samples with 51 action categories. We use the 3-way train/test split as provided by the authors. Table 8.3 shows a comparison with other approaches. The methods from (Simonyan and Zisserman, 2014a) and (Wang and Schmid, 2013) perform better than ours, however, both require computation of dense per frame optical flow for each video. In addition, the method in (Wang and Schmid, 2013) also requires camera motion estimation. Figure 8.5 shows the confusion matrix accumulated for all three splits. It can be seen that similar actions such as "throw" and "swing baseball" are the most confused. Figure 8.7 shows the top-5 predictions for selected test sequences.

Figure 8.7: Top-5 predictions using our approach for selected test sequences from the HMDB dataset (Kuehne et al., 2011) with 51 action categories. First row (green color) shows the ground-truth followed by predictions in decreasing level of confidence. Blue and red show correct and incorrect predictions, respectively.

8.5.5 Qualitative analysis

Since we do not preprocess the data using techniques such as background subtraction or tracking a bounding box, our feature-learning approach is agnostic to such domain-specific information. For this reason, wrong labels can be seen, in Figs. 8.6 and 8.7, when different activities are performed in visually similar environments. For example, Fig. 8.6(c6) vs. Fig. 8.6(b3) and Fig. 8.6(b6) vs. Fig. 8.6(c2), share similar environments and we see a high confidence of "HairCut" in the "ShavingBeard" action and "PlayingFlute" got confused with "PlayingViolin". Similar observations can be made in Fig. 8.7(b3) vs. Fig. 8.7(c6). However, sometimes background plays an important role in correctly recognizing certain actions, for instance, "SkyDiving" (Fig. 8.6(e3)) and "Surfing" (Fig. 8.6(e4)).

Other than similar background, actions may themselves be also visually confusing, which can affect feature learning. For example, Fig. 8.7(a2) vs. Fig. 8.7(c1). Both activities "cartwheel" and "handstand" entail performing a similar motion.

These are inherent problems of feature learning when using only raw data and the resulting mislabelings have been named *reasonable mistakes* in (Karpathy et al., 2014).

8.6 Summary and Future Work

We tackled the problem of action recognition by using a spatio-temporal feature learning scheme. This scheme allowed us to exploit the record-breaking pretrained single image classification model (Simonyan and Zisserman, 2014b). We report extensive experimental evaluations using challenging action recognition datasets. Our results are competitive with the state-of-the-art convolutional and strong feature-based baselines.

We are using the publicly available Torch7 library for our implementation which is optimized for fast processing on a CPU as well as a GPU. In our timing experiments we found that for a batch size of 60 videos, it takes ~ 1.6 seconds on a modern GPU to perform a single forward and backward pass through the network in Fig. 8.1 and about 6 hours (14 epochs) to train on complete UCF-101 training set. We expect to get further speed up by a more efficient implementation of 3D convolution.

So far, we concatenated the feature maps in the last convolutional layer. In the future, we plan to explore possible modifications in the network design to further exploit learning in the temporal domain. One possibility would be to gradually increase the number of temporal connections along the sequence of layers. This kind of adaptation is proposed in (Karpathy et al., 2014). We also plan to investigate the effect on performance of gradually clipping the top layers of the network and evaluation on the recently introduced Sports-1M dataset (Karpathy et al., 2014) which contains over 1 million labeled sample videos.

Chapter 9

Conclusions

The last decade has showed rapid developments in depth sensing technology coupled with significant increase in computing power from the graphics processing units (GPUs). This has led the computer vision community to successfully tackle some of the most challenging problems while encountering new ones. Our thesis is arguably an example of tackling such problems. We started using depth data for the task of segmentation and tracking. Afterwards, we resorted to higher-level tasks, which included object recognition, semantic segmentation and action recognition, where we developed methods improving over the state-of-the-art. To recap, we outline our main contributions as follows:

1. We proposed and evaluated a novel depth video segmentation approach. We used quadratic surface models that were adapted according to the changing input data. Our globally defined surface models were demonstrated to be more robust to artificially added noise when compared to other approaches. We also defined a depth-image-based affine motion tracker and made it more robust by combining it with our video segmentation approach.
2. We built a system which used the tracked results to chase and grasp a moving object in realtime. The system consisted of a WAM arm with 7-DoF. We used a robust inverse kinematics algorithm to continuously update the joint angles of the WAM arm until it grasped the tacked object.
3. We presented a new framework for object recognition. The framework consisted of a graphical model that exploited the geometric features of indoor scenes. Our model showed significant benefits in terms of computational speed as compared to another state-of-the-art approach.
4. For the semantic segmentation of indoor scenes, we proposed a unique feature that we called distance-from-wall. Using this feature in our deep learning model we showed an improvement in over all results.
5. Finally, we addressed the problem of video-content-based action recognition. We presented a unique convnet architecture that learned temporal features at a high level of abstraction. After combining our network with another trained model, we obtained state-of-the-art results on widely used benchmarking datasets.

9.1 Future Work

The task of building a robot that acts autonomously in complex environments is a big challenge which can be simplified by dividing and focusing on smaller parts. High-level problems that cover the different aspects of robot interaction can greatly benefit from the techniques introduced in this thesis. The work of semantic segmentation and action recognition is of particular significance and can be developed further to obtain a deeper insight of the scenes.

The unique approach to coherent depth video segmentation and tracking in Chapters 3, 4 and 5 opens up a new set of possibilities of extracting task relevant information, particularly from textureless surfaces. Another aspect to explore is the usage of segmentation for video communication and compression.

Our approach to object recognition in Chapter 6 showed promising results for two datasets, i.e., Cornell RGBD Dataset and Oakland Point Cloud Dataset which contain 17 and 5 object classes, respectively. It needs to be studied, how well our approach will generalize when there is a larger number of object classes.

The semantic image segmentation in Chapter 7 proved to be a good prior for improving an object discovery approach. This also opens up the possibility of improvement in other computer vision tasks such as object recognition and content-based image search. Extension of our approach to the video domain is another aspect that needs to be investigated.

Finally in Chapter 8, we revamped the idea of learning in the temporal domain by performing 3D convolution at a higher level of abstraction. However, the generalization performance of our model needs to be tested more thoroughly with larger databases such as the Sports-1M dataset (Karpathy et al., 2014).

9.2 Current Trends in Perception

Perception of human assistive environments has become a thriving field. Many innovative ideas are now emerging in flagship robotics, vision and machine learning conferences. In this work, we embarked on a journey of providing better ways to solve perception problems. During this journey, we saw some exciting developments in computer vision coupled with techniques from machine learning.

Some interesting ideas sprung with the introduction of the Kinect sensor for quick depth acquisition. For instance, histogram of oriented 4D normals from depth videos (Oreifej and Liu, 2013), anticipation of future activities (Koppula and Saxena, 2015) and object recognition by modeling scene context in 3D (Anand et al., 2013). The work of Anand et al. (2013) led us to build a more computationally efficient model for semantic object recognition.

Alongside the advances in depth sensing technology, standard benchmarks for image and video content recognition got redefined in terms of scalability from few hundreds to millions (Deng et al., 2009; Karpathy et al., 2014). Tackling such challenges required complete rethinking of the prevailing methods as conventional approaches using handcrafted features performed poorly on bigger datasets. Recent breakthroughs in deep learning are the consequence of such efforts. The ground breaking work of Krizhevsky et al. (2012), surpassing the previous records for single image classification used a deep convnet architecture. This work resulted in a paradigm shift in computer vision

practices using machine learning and many computer vision problems were resultantly tackled following this idea. The learned features were discovered to be highly generic and researchers started to solve different computer vision problems by transferring image representations learned from one problem to another (Oquab et al., 2014). For example, Eigen and Fergus (2015) used the ImageNet trained weights for predicting depths, surface normals and semantic labels. In line with such developments, we proposed our own semantic segmentation and action recognition with pretrained weights.

It is noteworthy that the underlying concepts and usage of deep convnets are not new. Some of the applications using convnets date back to the early 90s. Consider for example the work on object localization in images (Vaillant et al., 1994). The recent wave in the usage of deep learning techniques is greatly attributed to the availability of large amounts of data, increase in computing power, and publicly available GPU-optimized implementations such as Torch (Collobert et al., 2011), Caffe (Jia et al., 2014) and Theano (Bergstra et al., 2010).

Bibliography

Abramov, A., Pauwels, K., Papon, J., Worgotter, F., and Dellen, B. (2012). Real-time segmentation of stereo videos on a portable system with a mobile gpu. *IEEE Transactions on Circuits and Syst. for Video Technol.*, 22(9):1292–1305. 32, 33, 36, 38

Agostini, A., Torras, C., and Wörgötter, F. (2011). Integrating task planning and interactive learning for robots to work in human environments. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2386–2391. 38

Aksoy, E. E., Abramov, A., Dörr, J., Ning, K., Dellen, B., and Wörgötter, F. (2011). Learning the semantics of object action relations by observation. *Int. Journal of Robotics Research (IJRR)*, 30(10):1229–1249. 18, 19, 89

Allen, P., Timcenko, A., Yoshimi, B., and Michelman, P. (1993). Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165. 40

Anand, A., Koppula, H. S., Joachims, T., and Saxena, A. (2013). Contextually guided semantic labeling and search for three-dimensional point clouds. *Int. Journal of Robotics Research (IJRR)*, 32(1):19–34. 31, 64, 65, 66, 68, 69, 70, 71, 73, 88, 100

Anguelov, D., Taskarf, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative learning of markov random fields for segmentation of 3d scan data. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 169–176. 17

Apolloni, B., Ghosh, A., Alpaslan, F., and Patnaik, S. (2005). *Machine Learning and Robot Perception*, volume 7. Springer, 1st edition. 2

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2009). From contours to regions: An empirical evaluation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2294–2301. 22

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916. 17, 31

Archibald, C. and van de Panne, M. (1991). Tracking and grasping moving objects using reflex behaviour. In *Fifth International Conference on Advanced Robotics*, volume 1, pages 643–648. 40

Bab-Hadiashar, A. and Gheissari, N. (2006). Range image segmentation using surface selection criterion. *IEEE Transactions on Image Process.*, 15(7):2006–2018. 16, 29, 30

Babenko, B., Yang, M., and Belongie, S. (2011). Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(8):1619–1632. 52

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2012). Spatio-temporal convolutional sparse auto-encoder for sequence classification. In *British Machine Vision Conf. (BMVC)*, pages 124.1–124.1. 89

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359. 89

Benameur, K. and Belanger, P. (1998). Grasping of a moving object with a robotic hand-eye system. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 304–310. 40, 41

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, pages 3–10. 101

Bertalmio, M., Sapiro, G., and Randall, G. (2000). Morphing active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(7):733–737. 3

Bertozzi, M., Broggi, A., Del Rose, M., Felisa, M., Rakotomamonjy, A., and Suard, F. (2007). A pedestrian detector using histograms of oriented gradients and a support vector machine classifier. In *IEEE Intelligent Transportation Systems Conference*, pages 143–148. 11

Besl, P. and McKay, H. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256. 52

Bing, W. and Xiang, L. (2008). A simulation research on 3d visual servoing robot tracking and grasping a moving object. In *15th International Conference on Mechatronics and Machine Vision in Practice*, pages 362–367. 40, 41

Bober, M. (2001). Mpeg-7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716–719. 3

Bonnet, N., Cutrona, J., and Herbin, M. (2002). A 'no-threshold' histogram-based image segmentation method. *Pattern Recognition*, 35(10):2319 – 2322. 3

Bottou, L. (1998). Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK. revised, Oct. 2012. 67

Brown (2015). Brown university dataset. http://www.dam.brown.edu/ptg/brid/range/index.html. [Online; last accessed 29-Jan-2016]. 23

Cadena, C. and Kosecka, J. (2014). Semantic segmentation with heterogeneous sensor coverages. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2639–2645. 77

Chan, J. and Nejat, G. (2011). A learning-based control architecture for an assistive robot providing social engagement during cognitively stimulating activities. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 3928–3933. 87

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine*, 13(4):82–90. 39

Checchin, P., Trassoudaine, L., and Alizon, J. (1997). Segmentation of range images into planar regions. In *Proc. of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 156–163. 17, 29, 30

Chen, E., Xu, Y., Yang, X., and Zhang, W. (2013). Quaternion based optical flow estimation for robust object tracking. *Digital Signal Processing*, 23(1):118 – 125. 52

Choi, M. J., Torralba, A., and Willsky, A. (2012). A tree-based context model for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):240–252. 63

Chrungoo, A., Manimaran, S., and Ravindran, B. (2014). Activity recognition for natural human robot interaction. In *Social Robotics*, volume 8755 of *Lecture Notes in Computer Science*, pages 84–94. 87

Collet, A., Srinivasa, S. S., and Hebert, M. (2011). Structure discovery in multi-modal data: A region-based approach. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 5695–5702. 17

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *NIPS Workshop BigLearn*. 101

Colomé, A. and Torras, C. (2012). Redundant inverse kinematics: Experimental comparative review and two enhancements. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5333–5340. 40, 43

Colomé, A. and Torras, C. (2015). Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements. *IEEE/ASME Transactions on Mechatronics*, 20(2):944–955. 40, 43

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(5):564–577. 3

Couprie, C., Farabet, C., Najman, L., and LeCun, Y. (2013). Indoor semantic segmentation using depth information. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–8. 82, 84, 85, 86

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. 9

De Laet, T., Bellens, S., Smits, R., Aertbelien, E., Bruyninckx, H., and De Schutter, J. (2013). Geometric relations between rigid bodies (part 1): Semantics for standardization. *IEEE Robotics Automation Magazine*, 20(1):84–93. 64

Dellen, B., Husain, F., and Torras, C. (2013). Joint segmentation and tracking of object surfaces along human/robot manipulations. In *Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, volume 1, pages 244–251. 4, 51, 52, 59, 73

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. 100

Deng, Y. and Manjunath, B. (2001). Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(8):800–810. 18

Deselaers, T., Keysers, D., and Ney, H. (2008). Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107. 3

Dhanachandra, N., Manglem, K., and Chanu, Y. J. (2015). Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771. 3

DLR (2015). DLR dataset. http://www.robotic.dlr.de/242. [Online; last accessed 29-Jan-2016]. 27

Donahue, J., Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., and Saenko, K. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634. 93

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *Int. Conf. on Machine Learning (ICML)*. 77, 88

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208. 42, 59

Dragan, A., Ratliff, N., and Srinivasa, S. (2011). Manipulation planning with goal sets using constrained trajectory optimization. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 4582–4588. 75

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Int. Conf. on Computer Vision (ICCV)*, pages 2650–2658. 76, 77, 78, 84, 85, 86, 101

Enjarini, B. and Graser, A. (2012). Planar segmentation from depth images using gradient of depth feature. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4668–4674. 17, 29, 30

Fan, J., Zeng, G., Body, M., and Hacid, M. S. (2005). Seeded region growing: an extensive and comparative study. *Pattern Recogn. Letters*, 26(8):1139–1156. 52

Farabet, C., Couprie, C., Najman, L., and Lecun, Y. (2012). Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *Int. Conf. on Machine Learning (ICML)*, pages 575–582. 78

Foix, S., Alenyà, G., Cetto, J. A., and Torras, C. (2010). Object modeling using a tof camera under an uncertainty reduction approach. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1306–1312. 52

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions Math. Softw.*, 3(3):209–226. 58

Frigui, H. and Krishnapuram, R. (1999). A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):450–465. 17, 29, 30

Ganapathi, V., Plagemann, C., Thrun, S., and Koller, D. (2010). Real time motion capture using a single time-of-flight camera. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 755–762. 52

Ge, L. and Jie, Z. (2007). A real-time stereo visual servoing for moving object grasping based parallel algorithms. In *2nd IEEE Conference on Industrial Electronics and Applicat.*, pages 2886–2891. 40

Ghobadi, S., Loepprich, O., Hartmann, K., and Loffeld, O. (2007). Hand segmentation using 2d/3d images. In *Proc. Image and Vision Computing New Zealand*, pages 64–69. 15

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. 77, 88

Gold, S., Rangarajan, A., Lu, C.-P., Pappu, S., and Mjolsness, E. (1998). New algorithms for 2d and 3d point matching: pose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031. 52

Goodrich, M. A. and Schultz, A. C. (2007). Human-robot interaction: a survey. *Found. Trends Hum.-Comput. Interact.*, 1(3):203–275. 40

Gotardo, P., Bellon, O., Boyer, K., and Silva, L. (2004). Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm. *IEEE Transactions on Sys., Man, and Cybern., Part B: Cybern.*, 34(6):2303–2316. 17, 29, 30

Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *Int. Conf. on Computer Vision (ICCV)*, pages 1–8. 3

Granger, S. and Pennec, X. (2002). Multi-scale EM-ICP: A fast and robust approach for surface registration. In *Europ. Conf. on Computer Vision (ECCV)*, volume 2353, pages 418–432. 52

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2141–2148. 18, 32, 33, 36, 45, 58

Gupta, S., Girshick, R., Arbelaez, P., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. In *Europ. Conf. on Computer Vision (ECCV)*, volume 8695, pages 345–360. 76, 77, 79

Han, F., Tu, Z., and Zhu, S.-C. (2004). Range image segmentation by an effective jump-diffusion method. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9):1138–1153. 16, 23

Han, J., Volz, R., and Mudge, T. (1987). Range image segmentation and surface parameter extraction for 3-d object recognition of industrial parts. In *Int. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 380–386. 15

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Int. Conf. on Computer Vision (ICCV)*, pages 1026–1034. 3

He, X., Zemel, R., and Carreira-Perpindn, M. (2004). Multiscale conditional random fields for image labeling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 695–702. 64, 65

He, X. and Zemel, R. S. (2009). Learning hybrid models for image annotation with partially labeled data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 625–632. 3

Hegde, G. and Ye, C. (2011). A recursive planar feature extraction method for 3d range data segmentation. In *IEEE International Conference on Sys., Man, and Cybern.*, pages 3119–3124. 15

Hermans, A., Floros, G., and Leibe, B. (2014). Dense 3d semantic mapping of indoor scenes from rgb-d images. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2631–2638. 77, 78, 85, 86

Hirota, K. and Watanabe, H. (1990). Grasping 2d irregularly moving object using fuzzy controlled arm robot. In *First International Symposium on Uncertainty Modeling and Analysis*, pages 91–95. 40

Höft, N., Schulz, H., and Behnke, S. (2014). Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks. In *KI 2014: Advances in Artificial Intelligence*, volume 8736 of *Lecture Notes in Computer Science*, pages 80–85. 78

Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P., Bunke, H., Goldgof, D., Bowyer, K., Eggert, D., Fitzgibbon, A., and Fisher, R. (1996). An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(7):673–689. 16, 17, 22, 29, 30

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642. 42

Hu, N., Englebienne, G., Lou, Z., and Krose, B. (2014). Learning latent structure for activity recognition. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1048–1053. 88

Huang, T. and Netravali, A. (1994). Motion and structure from feature correspondences: a review. *Proceedings of the IEEE*, 82(2):252–268. 52

Husain, F., Colomé, A., Dellen, B., Alenya, G., and Torras, C. (2014a). Realtime tracking and grasping of a moving object from range video. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2617–2622. 5, 73, 75, 87

Husain, F., Dellen, B., and Torras, C. (2014b). Recognizing point clouds using conditional random fields. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 4257–4262. 5, 77

Husain, F., Dellen, B., and Torras, C. (2014c). Robust surface tracking in range image sequences. *Digital Signal Processing*, 35:37–44. 5, 88

Husain, F., Dellen, B., and Torras, C. (2015). Consistent depth video segmentation using adaptive surface models. *IEEE Transactions on Cybernetics*, 45(2):266–278. 4, 51, 52, 80, 81, 83

Husain, F., Dellen, B., and Torras, C. (2016a). Action recognition based on efficient deep feature learning in the spatio-temporal domain. *IEEE Robotics and Automation Letters (RA-L)*, 1(2):984–991. 6, 37

Husain, F., Schulz, H., Dellen, B., Torras, C., and Behnke, S. (2016b). Combining semantic and geometric features for object class segmentation of indoor scenes. *IEEE Robotics and Automation Letters (RA-L)*, 2(1):49–55. 5, 65

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computing*, 25(2):328–373. 43

Isard, M. and Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *Int. Journal of Computer Vision (IJCV)*, 29(1):5–28. 3

Jaderberg, M., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*. 88

Jain, A., Tompson, J., LeCun, Y., and Bregler, C. (2014). Modeep: A deep learning framework using motion features for human pose estimation. In *Asian Conference on Computer Vision*, pages 302–315. 88

Jain, M., Jegou, H., and Bouthemy, P. (2013). Better exploiting motion for better action recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2555–2562. 95

Jepson, A., Fleet, D., and El-Maraghi, T. (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(10):1296–1311. 55

Jhuang, H., Serre, T., Wolf, L., and Poggio, T. (2007). A biologically inspired system for action recognition. In *Int. Conf. on Computer Vision (ICCV)*, pages 1–8. 95

Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):221–231. 89

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*. 88, 101

Jiang, X. (2000). An adaptive contour closure algorithm and its experimental evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(11):1252–1265. 17, 29, 30

Jiang, X., Bowyer, K., Morioka, Y., Hiura, S., Sato, K., Inokuchi, S., Bock, M., Guerra, C., Loke, R., and du Buf, J. (2000a). Some further results of experimental comparison of range image segmentation algorithms. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 877 –881. 17, 29, 30

Jiang, X., Bunke, H., and Meier, U. (2000b). High-level feature based range image segmentation. *Image and Vision Computing*, 18(10):817 – 822. 17

Jiang, X., Hofer, S., Stahs, T., Ahrns, I., and Bunke, H. (1999). Extraction and tracking of surfaces in range image sequences. In *International Conference on 3-D Digital Imaging and Modeling*, pages 252–260. 15, 18, 51, 52

Jiang, Y.-G., Dai, Q., Xue, X., Liu, W., and Ngo, C.-W. (2012). Trajectory-based modeling of human actions with motion reference points. In *Europ. Conf. on Computer Vision (ECCV)*, volume 7576, pages 425–438. 95

Kanda, T., Shiomi, M., Miyashita, Z., Ishiguro, H., and Hagita, N. (2010). A communication robot in a shopping mall. *IEEE Transactions on Robotics*, 26(5):897–913. 87

Karaman, S., , Seidenari, L., Bagdanov, A., and Bimbo, A. (2013). L1-regularized logistic regression stacking and transductive crf smoothing for action recognition in video. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, pages 1–6. 93

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732. 88, 89, 90, 92, 93, 94, 95, 97, 98, 100

Khan, S., Bennamoun, M., Sohel, F., and Togneri, R. (2014). Geometry driven semantic labeling of indoor scenes. In *Europ. Conf. on Computer Vision (ECCV)*, volume 8689 of *Lecture Notes in Computer Science*, pages 679–694. 84

Kim, D. and Kim, D. (2010). A fast icp algorithm for 3-d human body motion tracking. *IEEE Signal Process. Lett.*, 17(4):402–405. 51

Klaeser, A., Marszalek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conf. (BMVC)*, pages 99.1–99.10. 88, 89

Knoop, S., Vacek, S., and Dillmann, R. (2006). Sensor fusion for 3d human body tracking with an articulated 3d body model. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1686 –1691. 52

Kondak, K., Binner, S., Hommel, G., and Neumann, M. (2001). Time optimal manipulator control for sensor guided grasping of moving objects. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 4, pages 1912–1917. 40, 41

Koppula, H. and Saxena, A. (2015). Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, PP(99):1–1. 100

Koster, K. and Spann, M. (2000). Mir: an approach to robust clustering-application to range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(5):430–444. 17, 29, 30

Krainin, M., Henry, P., Ren, X., and Fox, D. (2011). Manipulator and object tracking for in-hand 3d object modeling. *Int. Journal of Robotics Research (IJRR)*, 30(11):1311–1327. 52

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc. 78, 91, 93, 100

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). HMDB: a large video database for human motion recognition. In *Int. Conf. on Computer Vision (ICCV)*, pages 2556–2563. 93, 96, 97

Kulpate, C., Mehrandezh, M., and Paranjape, R. (2005). An eye-to-hand visual servoing structure for 3d positioning of a robotic arm using one camera and a flat mirror. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1464–1470. 39

Kumar, S. and Hebert, M. (2005). A hierarchical field framework for unified context-based classification. In *Int. Conf. on Computer Vision (ICCV)*, pages 1284–1291. 65

Kwon, J., Lee, K. M., and Park, F. (2009). Visual tracking via geometric particle filtering on the affine group with optimal importance functions. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 991–998. 3, 9, 39, 41, 42, 44, 45, 50, 52, 55, 57, 59, 60

Kwon, J. and Park, F. C. (2010). Visual tracking via particle filtering on the affine group. *Int. Journal of Robotics Research (IJRR)*, 29(2-3):198–217. 42, 50

Ladicky, L., Russell, C., Kohli, P., and Torr, P. (2009). Associative hierarchical crfs for object class image segmentation. In *Int. Conf. on Computer Vision (ICCV)*, pages 739–746. 3

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1817–1824. 26, 28, 33

Lai, K., Bo, L., Ren, X., and Fox, D. (2012). Detection-based object labeling in 3d scenes. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1330–1337. 64, 65

Lakaemper, R. and Latecki, L. (2006). Using extended em to segment planar structures in 3d. In *Int. Conf. on Pattern Recognition (ICPR)*, volume 3, pages 1077–1082. 16

Laptev, I. (2005). On space-time interest points. *Int. Journal of Computer Vision (IJCV)*, 64(2-3):107–123. 88

Le, Q., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. (2011a). On optimization methods for deep learning. In *Int. Conf. on Machine Learning (ICML)*, pages 265–272. 89, 92

Le, Q., Zou, W., Yeung, S., and Ng, A. (2011b). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3361–3368. 88, 89

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 13, 88

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Int. Conf. on Machine Learning (ICML)*, pages 609–616. 89

Lei, M. and Ghosh, B. (1993). Visually guided robotic tracking and grasping of a moving object. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, volume 2, pages 1604–1609. 40, 41

Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011). Pylon model for semantic segmentation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1485–1493. 3

Leonardis, A., Jaklic, A., and Solina, F. (1997). Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(11):1289–1295. 16, 17

Li, C., Saxena, A., and Chen, T. (2011). θ-mrf: Capturing spatial and semantic structure in the parameters for scene understanding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 549–557. 64

Li, M., Tan, T., Chen, W., and Huang, K. (2012). Efficient object tracking by incremental self-tuning particle filtering on the affine group. *IEEE Transactions on Image Processing*, 21(3):1298–1313. 10

Li, P., Chaumette, F., and Tahri, O. (2005). A shape tracking algorithm for visual servoing. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2847–2852. 39

Li, X., Hu, W., Zhang, Z., Zhang, X., and Luo, G. (2007). Robust visual tracking based on incremental tensor subspace learning. In *Int. Conf. on Computer Vision (ICCV)*, pages 1–8. 42

Liu, R., Zhang, X., Webb, J., and Li, S. (2015). Context-specific intention awareness through web query in robotic caregiving. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1962–1967. 87

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. 76, 78

Lopez-Mendez, A., Alcoverro, M., Pardas, M., and Casas, J. (2011). Real-time upper body tracking with online initialization using a range sensor. In *IEEE International Conference on Comput. Vision Workshops*, pages 391–398. 18

Maeda, K., Minami, M., Yanou, A., Matsumoto, H., Yu, F., and Hou, S. (2012). Frequency response experiments of 3-d pose full-tracking visual servoing with eye-vergence hand-eye robot system. In *Proceedings of SICE Annual Conference*, pages 101–107. 39

Malis, E., Chaumette, F., and Boudet, S. (1999). 2-1/2 D visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250. 39

Malisiewicz, T. and Efros, A. (2009). Beyond categories: The visual memex model for reasoning about object relationships. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1222–1230. 65

Martín García, G., Husain, F., Schulz, H., Frintrop, S., Torras, C., and Behnke, S. (2016). Semantic segmentation priors for object discovery. In *Int. Conf. on Pattern Recognition (ICPR)*. Accepted. 5, 86

Martín García, G., Potapova, E., Werner, T., Zillich, M., Vincze, M., and Frintrop, S. (2015). Saliency-based Object Discovery on RGB-D Data with a Late-Fusion Approach. *Int. Conf. on Robotics and Automation (ICRA)*, pages 1866–1873. 86

Martínez, D., Alenyà, G., and Torras, C. (2015). Planning robot manipulation to clean planar surfaces. *Engineering Applications of Artificial Intelligence*, 39:23–32. 75

Min, J., Powell, M., and Bowyer, K. (2004). Automated performance evaluation of range image segmentation algorithms. *IEEE Transactions on Syst., Man, and Cybern., Part B: Cybern.*, 34(1):263–271. 16

Müller, A. and Behnke, S. (2014). Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 6232–6237. 77, 84

Munoz, D. (2013). *Inference Machines: Parsing Scenes via Iterated Predictions*. PhD thesis. 69

Munoz, D., Bagnell, J. A. D., Vandapel, N., and Hebert, M. (2009a). Contextual classification with functional max-margin markov networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 975–982. 65, 68, 69, 72, 73

Munoz, D., Vandapel, N., and Hebert, M. (2009b). Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2009–2016. 68

Murphy, K., Torralba, A., and Freeman, W. T. (2004). Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1499–1506. 64

Murthy, O. R. and Goecke, R. (2013). Combined ordered and improved trajectories for large scale human action recognition. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, pages 412–419. 93

Nakagawa, S., Di, P., Hasegawa, Y., Fukuda, T., Kondo, I., Tanimoto, M., and Huang, J. (2015). Tandem stance avoidance using adaptive and asymmetric admittance control for fall prevention. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 5898–5903. 87

Oehler, B., Stückler, J., Welle, J., Schulz, D., and Behnke, S. (2011). Efficient multi-resolution plane segmentation of 3d point clouds. In *4th International Conference on Intelligence Robotics and Applicat.*, pages 145–156. 17, 29, 30, 31

Oikonomidis, I., Kyriazis, N., and Argyros, A. (2012). Tracking the articulated motion of two strongly interacting hands. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1862–1869. 52

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 88, 101

Oreifej, O. and Liu, Z. (2013). Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 716–723. 100

Ouyang, W., Chu, X., and Wang, X. (2014). Multi-source deep learning for human pose estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2344. 88

Papon, J., Abramov, A., Aksoy, E., and Worgotter, F. (2012). A modular system architecture for online parallel vision pipelines. In *IEEE Workshop on Applicat. of Comput. Vision*, pages 361–368. 32, 33, 36

Park, K.-H., Lee, H.-E., Kim, Y., and Bien, Z. (2008). A steward robot for human-friendly human-machine interaction in a smart house environment. *IEEE Transactions on Automation Science and Engineering*, 5(1):21–25. 87

Park, M., Jin, J., and Wilson, L. (2002). Fast content-based image retrieval using quasi-gabor filter and reduction of image feature dimension. In *IEEE Southwest Symposium on Image Analysis and Interpretation*. 3

Parvizi, E. and Wu, Q. (2008). Multiple object tracking based on adaptive depth segmentation. In *Canadian Conference on Comput. and Robot Vision*, pages 273–277. 15, 16, 18

Patras, I., Hendriks, E., and Lagendijk, R. (2001). Video segmentation by map labeling of watershed segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(3):326–332. 18

Pauwels, K., Rubio, L., Diaz Alonso, J., and Ros, E. (2013). Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2347–2354. 18

Pieropan, A., Ek, C., and Kjellstrom, H. (2013). Functional object descriptors for human activity modeling. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1282–1289. 18

Pinheiro, P. and Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *Int. Conf. on Machine Learning (ICML)*, pages 82–90. JMLR Workshop and Conference Proceedings. 88

Powell, M., Bowyer, K., Jiang, X., and Bunke, H. (1998). Comparing curved-surface range image segmenters. In *Int. Journal of Computer Vision (IJCV)*, pages 286–291. 16

Pulli, K. and Pietikainen, M. (1993). Range image segmentation based on decomposition of surface normals. In *8th Scandinavian Conference on Image Analysis*. 15

Quattoni, A., Collins, M., and Darrell, T. (2004). Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1104. 63, 65

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Int. Conf. on Computer Vision (ICCV)*, pages 1–8. 65

Ramisa, A., Alenya, G., Moreno-Noguer, F., and Torras, C. (2013). Finddd: A fast 3d descriptor to characterize textiles for robot manipulation. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 824–830. 64

Ren, X., Bo, L., and Fox, D. (2012). Rgb-(d) scene labeling: Features and algorithms. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2759–2766. 17, 31, 37

Rhee, S.-M., Lee, Y.-B., Kim, J. D. K., and Rhee, T. (2012). Split and merge approach for detecting multiple planes in a depth image. In *19th IEEE International Conference on Image Processing*, pages 1213–1216. 16

Ross, D., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *Int. Journal of Computer Vision (IJCV)*, 77(1):125–141. 42, 44, 45, 57, 58, 59, 60

Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. (2006). 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. Journal of Computer Vision (IJCV)*, 66(3):231–259. 63

Rozo, L., Jiménez, P., and Torras, C. (2013). A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent Service Robotics*, 6(1):33–51. 38

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *International Conference on 3D Digital Imaging and Modeling*, pages 145–152. 52, 58

Rusu, R., Blodow, N., Marton, Z., and Beetz, M. (2008a). Aligning point cloud views using persistent feature histograms. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3384–3391. 9, 12

Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008b). Persistent point feature histograms for 3d point clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, pages 119–128, Baden-Baden, Germany. 64

Sabata, B. and Aggarwal, J. K. (1996). Surface correspondence and motion computation from a pair of range images. *Comput. Vision and Image Understanding*, 63(2):232–250. 51, 52

Sadanand, S. and Corso, J. (2012). Action bank: A high-level representation of activity in video. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1234–1241. 89

Sandhu, R., Dambreville, S., and Tannenbaum, A. (2010). Point set registration via particle filtering and stochastic dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32:1459–1473. 51, 52

Savarese, S. and Fei-Fei, L. (2007). 3d generic object categorization, localization and pose estimation. In *IEEE 11th International Conference on Computer Vision*, pages 1–8. 64

Saxena, A., Schulte, J., and Ng, A. Y. (2007). Depth estimation using monocular and stereo cues. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2197–2203. 2

Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Huijnen, C., van den Heuvel, H., van Berlo, A., Bley, A., and Gross, H.-M. (2013). Realization and user evaluation of a companion robot for people with mild cognitive impairments. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1153–1159. 87

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: a local svm approach. In *Int. Conf. on Pattern Recognition (ICPR)*, volume 3, pages 32–36. 88

Schulz, H., Höft, N., and Behnke, S. (2015). Depth and height aware semantic rgb-d perception with convolutional neural networks. In *Europ. Symposium on Artificial Neural Networks (ESANN)*. 76, 78

Schwarz, M., Schulz, H., and Behnke, S. (2015). RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1329–1335. 77

Schwing, A. G., Hazan, T., Pollefeys, M., and Urtasun, R. (2011). Distributed message passing for large scale graphical models. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1833–1840. 73

Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th International Conference on Multimedia*, pages 357–360. 88, 89

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–16. 9, 79

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 77

Shen, C., Kim, J., and Wang, H. (2010). Generalized kernel-based visual tracking. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(1):119–130. 3

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905. 3

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Europ. Conf. on Computer Vision (ECCV)*, pages 1–15. 65

Siciliano, B., Sciavicco, L., Oriolo, G., and Villani, L. (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing, Springer. 43

Silberman, N. and Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, pages 601–608. 76, 77, 81

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *Europ. Conf. on Computer Vision (ECCV)*, pages 746–760. 17, 31, 32, 37, 77, 82

Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576. 3, 88, 90, 93, 94, 95, 96

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556. 90, 91, 98

Siradjuddin, I., Behera, L., McGinnity, T., and Coleman, S. (2012). A position based visual tracking system for a 7 dof robot manipulator using a kinect camera. In *Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–7. 40, 41

Smith, C. and Papanikolopoulos, N. (1995). Grasping of static and moving objects using a vision-based control approach. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 329–334. 40, 41

Song, W., Minami, M., Yu, F., Zhang, Y., and Yanou, A. (2011). 3-d hand and eye-vergence approaching visual servoing with lyapunouv-stable pose tracking. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 5210–5217. 39

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402. 93, 94, 96

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681. 93

Stefan Mathe, C. S. (2012). Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *Europ. Conf. on Computer Vision (ECCV)*, volume 7573, pages 842–856. 88

Stückler, J., Waldvogel, B., Schulz, H., and Behnke, S. (2015). Dense real-time mapping of object-class semantics from rgb-d video. *Journal of Real-Time Image Processing*, pages 599–609. 84

Sung, J., Ponce, C., Selman, B., and Saxena, A. (2012). Unstructured human activity detection from rgbd images. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 842–849. 88

Tamaki, T., Abe, M., Raytchev, B., and Kaneda, K. (2010). Softassign and EM-ICP on gpu. In *International Conference on Networking and Computing*, pages 179–183. 53

Taylor, G. W., Fergus, R., LeCun, Y., and Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *Europ. Conf. on Computer Vision (ECCV)*, pages 140–153. 89

Tombari, F., Gori, F., and Di Stefano, L. (2011). Evaluation of stereo algorithms for 3d object recognition. In *IEEE International Conference on Computer Vision Workshops*, pages 990–997. 63

Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Int. Conf. on Computer Vision (ICCV)*, pages 4489–4497. 3, 9, 89, 90, 92, 93

Tsap, L. V. and Shin, M. C. (2004). Dynamic disparity adjustment and histogram-based filtering of range data for fast 3-d hand tracking. *Digital Signal Processing*, 14(6):550 – 565. 52

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 477–485. 67

USF (2015). Cesar lab at oak ridge national laboratory. http://marathon.csee.usf.edu/range/DataBase.html. [Online; last accessed 29-Jan-2016]. 27, 29

Vaillant, R., Monrocq, C., and Le Cun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image and Signal Processing*, 141(4):245–250. 101

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Int. Conf. on Machine Learning (ICML)*, pages 969–976. 72

Wallenberg, M., Felsberg, M., Forssén, P.-E., and Dellen, B. (2011). Channel Coding for Joint Colour and Depth Segmentation. In *Proceedings of Pattern Recognition 33rd DAGM Symposium*, volume 6835, pages 306–315. 17

Wang, C., de La Gorce, M., and Paragios, N. (2009a). Segmentation, ordering and multi-object tracking using graphical models. In *Int. Conf. on Computer Vision (ICCV)*, pages 747–754. 18, 19

Wang, C. and Liu, H. (2012). Robust visual tracking based on adaptive depth-color-cue integration using range sensor. In *IEEE Conference on Multisensor Fusion and Integration for Intelligence Syst.*, pages 330–335. 18

Wang, D. (1998). Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits and Syst. for Video Technol.*, 8(5):539–546. 18

Wang, H., Oneata, D., Verbeek, J., and Schmid, C. (2015). A robust and efficient video representation for action recognition. *Int. Journal of Computer Vision (IJCV)*, pages 1–20. 93

Wang, H. and Schmid, C. (2013). Action recognition with improved trajectories. In *Int. Conf. on Computer Vision (ICCV)*, pages 3551–3558. 3, 89, 93, 95, 96

Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. (2009b). Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conf. (BMVC)*, pages 124.1–124.11. 88

Wang, H., Zhou, H., and Finn, A. (2014). Discriminative dictionary learning via shared latent structure for object recognition and activity recognition. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 6299–6304. 88

Wang, X., Bai, X., Yang, X., Liu, W., and Latecki, L. J. J. (2011). Maximal cliques that satisfy hard constraints with application to deformable object model learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 864–872. 65

Weiss, Y. (2001). Comparing the mean field method and belief propagation for approximate inference in mrfs. In *Saad and Opper, eds., Advanced Mean Field Methods.* MIT press. 67

Willems, G., Tuytelaars, T., and Gool, L. (2008). An efficient dense and scale-invariant spatio-temporal interest point detector. In *Europ. Conf. on Computer Vision (ECCV)*, pages 650–663. 88, 89

Wolf, D., Prankl, J., and Vincze, M. (2015). Fast semantic segmentation of 3d point clouds using a dense crf with learned parameters. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 4867–4873. 77, 78, 84, 85, 86

Wu, C., Lenz, I., and Saxena, A. (2014). Hierarchical semantic labeling for task-relevant rgb-d perception. In *Robotics: Science and Systems (RSS)*, pages 1–9. 78

Xiong, X. and Huber, D. (2010). Using context to create semantic 3d models of indoor environments. In *British Machine Vision Conf. (BMVC)*, pages 1–11. 65

Xiong, X., Munoz, D., Bagnell, J., and Hebert, M. (2011). 3-d scene analysis via sequenced predictions over points and regions. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2609–2616. 77

Yan, H., Ang, M. H., and Poo, A. N. (2013). A survey on perception methods for human–robot interaction in social robots. *International Journal of Social Robotics*, 6(1):85–119. 2

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Understanding belief propagation and its generalizations. pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 67

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, 38(4). 52

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Europ. Conf. on Computer Vision (ECCV)*, pages 818–833. Springer International Publishing. 91

Zha, S., Luisier, F., Andrews, W., Srivastava, N., and Salakhutdinov, R. (2015). Exploiting image-trained cnn architectures for unconstrained video classification. In *British Machine Vision Conf. (BMVC)*, pages 60.1–60.13. 90

Zhang, H., Zhou, W., and Parker, L. (2014). Fuzzy segmentation and recognition of continuous human activities. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 6305–6312. 89

Zhang, R., Candra, S., Vetter, K., and Zakhor, A. (2015). Sensor fusion for semantic segmentation of urban scenes. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 1850–1857. 75, 77, 78

Zhou, S., Chellappa, R., and Moghaddam, B. (2004). Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13(11):1491–1506. 42